

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2020

Z. Ali
R. Gandhi
C. Filsfils
F. Brockners
N. Nainar
C. Pignataro
Cisco Systems, Inc.
C. Li
M. Chen
Huawei
G. Dawra
LinkedIn
November 3, 2019

Segment Routing Header encapsulation for In-situ OAM Data
draft-ali-spring-ioam-srv6-02

Abstract

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). IOAM records operational and telemetry information in the data packet while the packet traverses a path between two points in the network. This document defines how IOAM data fields are transported as part of the Segment Routing with IPv6 data plane (SRv6) header.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|---|
| 1. Introduction | 2 |
| 2. Conventions | 3 |
| 2.1. Requirement Language | 3 |
| 2.2. Abbreviations | 3 |
| 3. OAM Metadata Piggybacked in Data Packets | 4 |
| 3.1 IOAM Data Field Encapsulation in SRH | 4 |
| 4. Procedure | 5 |
| 4.1. Ingress Node | 5 |
| 4.2. SR Segment Endpoint Node | 5 |
| 4.3. Egress Node | 6 |
| 5. IANA Considerations | 6 |
| 6. Security Considerations | 6 |
| 7. Acknowledgements | 6 |
| 8. References | 7 |
| 8.1. Normative References | 7 |
| 8.2. Informative References | 7 |
| Authors' Addresses | 8 |

1. Introduction

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). IOAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the IOAM data fields are added to the data packets rather than being sent within probe packets specifically dedicated to OAM.

This document defines how IOAM data fields are transported as part of the Segment Routing with IPv6 data plane (SRv6) header [I-D.6man-segment-routing-header].

The IOAM data fields carried are defined in [I-D.ietf-ippm-ioam-data], and can be used for various use-cases including Performance Measurement (PM) and Proof-of-Transit (PoT).

2. Conventions

2.1. Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

Abbreviations used in this document:

IOAM In-situ Operations, Administration, and Maintenance

OAM Operations, Administration, and Maintenance

PM Performance Measurement

PoT Proof-of-Transit

SR Segment Routing

SRH SRv6 Header

SRv6 Segment Routing with IPv6 Data plane

3. OAM Metadata Piggybacked in Data Packets

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). This section describes IOAM functionality in SRv6 network.

The IOAM data is carried in SRH.TLV. This enables the IOAM mechanism to build on the network programmability capability of SRv6. Specifically, the ability for an SRv6 endpoint to determine whether to process or ignore some specific SRH TLVs is based on the SID function. This enables collection of the IOAM information hardware friendly based on the intermediate endpoint capability. The nodes that are not capable of supporting the IOAM functionality does not have to look or process SRH TLV (i.e., such nodes can simply ignore the SRH IOAM TLV). This also enable collection of IOAM data only from segment endpoint.

3.1 IOAM Data Field Encapsulation in SRH

The SRv6 encapsulation header (SRH) is defined in [I-D.ietf-6man-segment-routing-header]. IOAM data fields are carried in the SRH, using a single pre-allocated SRH TLV. The different IOAM data fields defined in [I-D.ietf-ippm-ioam-data] are added as sub-TLVs.

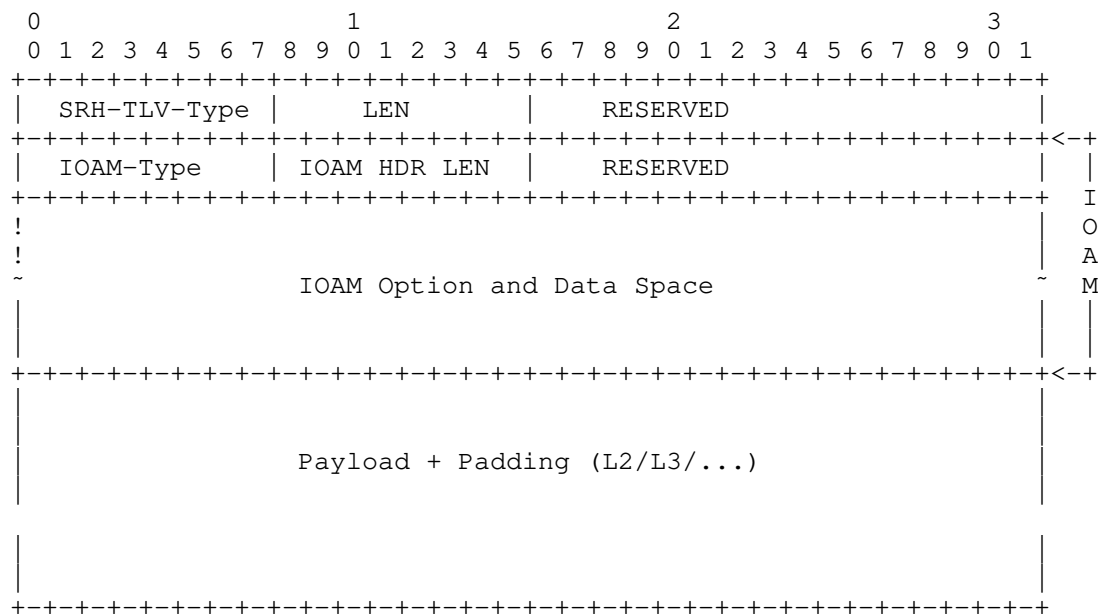


Figure 1: IOAM data encapsulation in SRH

SRH-TLV-Type: IOAM TLV Type for SRH is defined as TBA1.

The fields related to the encapsulation of IOAM data fields in the SRH are defined as follows:

IOAM-Type: 8-bit field defining the IOAM Option type, as defined in Section 7.2 of [I-D.ietf-ippm-ioam-data].

IOAM HDR LEN: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

RESERVED: 8-bit reserved field MUST be set to zero upon transmission and ignored upon receipt.

IOAM Option and Data Space: IOAM option header and data is present as defined by the IOAM-Type field, and is defined in Section 4 of [I-D.ietf-ippm-ioam-data].

4. Procedure

This section summarizes the procedure for IOAM data encapsulation in SRv6 SRH. The SR nodes implementing the IOAM functionality follows the MTU and other considerations outlined in [I-D.6man-extension-header-insertion].

4.1. Ingress Node

As part of the SRH encapsulation, the ingress node of an SR domain or an SR Policy [I-D.ietf-spring-segment-routing-policy] MAY add the IOAM TLV in the SRH of the data packet. If an ingress node supports IOAM functionality and, based on a local configuration, wants to collect IOAM data, it adds IOAM TLV in the SRH. Based on the size of the segment list (SL), the ingress node preallocates space in the IOAM TLV.

If IOAM data from the last node in the segment-list (Egress node) is desired, the ingress uses an Ultimate Segment Pop (USP) SID advertised by the Egress node.

The ingress node MAY also insert the IOAM data about the local information in the IOAM TLV in the SRH at index 0 of the preallocated IOAM TLV.

4.2. Intermediate SR Segment Endpoint Node

The SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is a local SID. As part of the SR Header processing as described in [I-D.ietf-6man-segment-routing-header] and [I-D.ietf-spring-srv6-network-programming], the SR Segment Endpoint node performs the following IOAM operations.

If an intermediate SR segment endpoint node is not capable of processing IOAM TLV, it simply ignores it. I.e., it does not have to look or process SRH TLV.

If an intermediate SR segment endpoint node is capable of processing IOAM TLV and the local SID supports IOAM data recording, it checks if any SRH TLV is present in the packet using procedures defined in [I-D.ietf-6man-segment-routing-header]. If the node finds IOAM TLV in the SRH it finds the local index at which it is expected to record the IOAM data. The local index is found using the SRH.SL field. The node records the IOAM data at the desired preallocated space.

4.3. Egress Node

The Egress node is the last node in the segment-list of the SRH. When IOAM data from the Egress node is desired, a USP SID advertised by the Egress node is used by the Ingress node.

The processing of IOAM TLV at the Egress node is similar to the processing of IOAM TLV at the SR Segment Endpoint Node. The only difference is that the Egress node may telemeter the IOAM data to an external entity.

5. IANA Considerations

IANA is requested to allocate a mutable SRH TLV Type for IOAM TLV data fields under registry name "Segment Routing Header TLVs" requested by [I-D.6man-segment-routing-header].

| SRH TLV Type | Description | Reference |
|-----------------------|--------------------------|---------------|
| TBA1 Greater than 128 | TLV for IOAM Data Fields | This document |

6. Security Considerations

The security considerations of SRv6 are discussed in [I-D.spring-srv6-network-programming] and [I-D.6man-segment-routing-header], and the security considerations of IOAM in general are discussed in [I-D.ietf-ippm-ioam-data].

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain.

7. Acknowledgements

The authors would like to thank Shwetha Bhandari and Vengada Prasad Govindan for the discussions on IOAM.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.
- [I-D.spring-srv6-network-programming] Filsfils, C. et al. "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming, work in progress.
- [I-D.6man-segment-routing-header] Previdi, S., Filsfils, C. et al, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header, work in progress.
- [I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and Bernier, D., "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data, work in progress.
- [I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

8.2. Informative References

- [I-D.6man-extension-header-insertion] D. Voyer, et al., "Insertion of IPv6 Segment Routing Headers in a Controlled Domain", draft-voyer-6man-extension-header-insertion, work in progress.

Authors' Addresses

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Frank Brockners
Cisco Systems, Inc.
Germany

Email: fbrockne@cisco.com

Nagendra Kumar Nainar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Carlos Pignataro
Cisco Systems, Inc.

Email: cpignata@cisco.com

Cheng Li
Huawei

Email: chengli13@huawei.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Gaurav Dawra
LinkedIn

Email: gdawra.ietf@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 12, 2022

Z. Ali
R. Gandhi
C. Filsfils
F. Brockners
N. Nainar
C. Pignataro
Cisco Systems, Inc.
C. Li
M. Chen
Huawei
G. Dawra
LinkedIn
January 12, 2022

Segment Routing Header encapsulation for In-situ OAM Data
draft-ali-spring-ioam-srv6-05

Abstract

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). IOAM records operational and telemetry information in the data packet while the packet traverses a path between two points in the network. This document defines how IOAM data fields are transported as part of the Segment Routing with IPv6 data plane (SRv6) header.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 12, 2022.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|---|
| 1. Introduction | 2 |
| 2. Conventions | 3 |
| 2.1. Requirement Language | 3 |
| 2.2. Abbreviations | 3 |
| 3. OAM Metadata Piggybacked in Data Packets | 4 |
| 3.1 IOAM Data Field Encapsulation in SRH | 4 |
| 4. Procedure | 5 |
| 4.1. Ingress Node | 5 |
| 4.2. SR Segment Endpoint Node | 5 |
| 4.3. Egress Node | 6 |
| 5. IANA Considerations | 6 |
| 6. Security Considerations | 6 |
| 7. Acknowledgements | 6 |
| 8. References | 7 |
| 8.1. Normative References | 7 |
| 8.2. Informative References | 7 |
| Authors' Addresses | 8 |

1. Introduction

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). IOAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the IOAM data fields are added to the data packets rather than being sent within probe packets specifically dedicated to OAM.

This document defines how IOAM data fields are transported as part of the Segment Routing with IPv6 data plane (SRv6) header [I-D.6man-segment-routing-header].

The IOAM data fields carried are defined in [I-D.ietf-ippm-ioam-data], and can be used for various use-cases including Performance Measurement (PM) and Proof-of-Transit (PoT).

2. Conventions

2.1. Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

Abbreviations used in this document:

| | |
|------|---|
| IOAM | In-situ Operations, Administration, and Maintenance |
| OAM | Operations, Administration, and Maintenance |
| PM | Performance Measurement |
| PoT | Proof-of-Transit |
| SR | Segment Routing |
| SRH | SRv6 Header |
| SRv6 | Segment Routing with IPv6 Data plane |

3. OAM Metadata Piggybacked in Data Packets

OAM and PM information from the SR endpoints can be piggybacked in the data packet. The OAM and PM information piggybacking in the data packets is also known as In-situ OAM (IOAM). This section describes IOAM functionality in SRv6 network.

The IOAM data is carried in SRH.TLV. This enables the IOAM mechanism to build on the network programmability capability of SRv6. Specifically, the ability for an SRv6 endpoint to determine whether to process or ignore some specific SRH TLVs is based on the SID function. This enables collection of the IOAM information hardware friendly based on the intermediate endpoint capability. The nodes that are not capable of supporting the IOAM functionality does not have to look or process SRH TLV (i.e., such nodes can simply ignore the SRH IOAM TLV). This also enable collection of IOAM data only from segment endpoint.

3.1 IOAM Data Field Encapsulation in SRH

The SRv6 encapsulation header (SRH) is defined in [I-D.ietf-6man-segment-routing-header]. IOAM data fields are carried in the SRH, using a single pre-allocated SRH TLV. The different IOAM data fields defined in [I-D.ietf-ippm-ioam-data] are added as sub-TLVs.

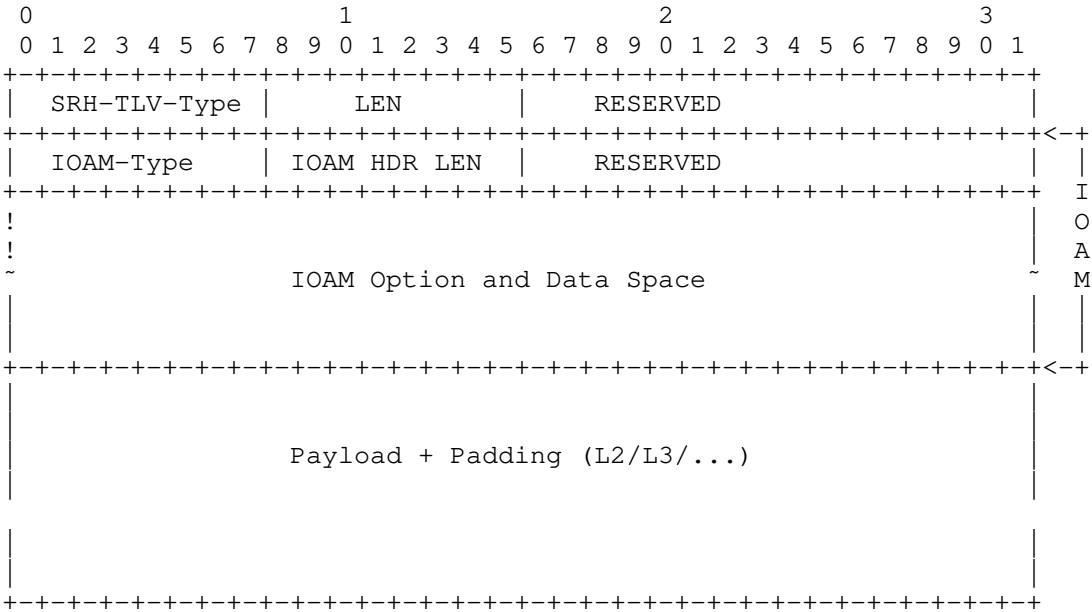


Figure 1: IOAM data encapsulation in SRH

SRH-TLV-Type: IOAM TLV Type for SRH is defined as TBA1.

The fields related to the encapsulation of IOAM data fields in the SRH are defined as follows:

IOAM-Type: 8-bit field defining the IOAM Option type, as defined in Section 7.2 of [I-D.ietf-ippm-ioam-data].

IOAM HDR LEN: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

RESERVED: 8-bit reserved field MUST be set to zero upon transmission and ignored upon receipt.

IOAM Option and Data Space: IOAM option header and data is present as defined by the IOAM-Type field, and is defined in Section 4 of [I-D.ietf-ippm-ioam-data].

4. Procedure

This section summarizes the procedure for IOAM data encapsulation in SRv6 SRH. The SR nodes implementing the IOAM functionality follows the MTU and other considerations outlined in [I-D.6man-extension-header-insertion].

4.1. Ingress Node

As part of the SRH encapsulation, the ingress node of an SR domain or an SR Policy [I-D.ietf-spring-segment-routing-policy] MAY add the IOAM TLV in the SRH of the data packet. If an ingress node supports IOAM functionality and, based on a local configuration, wants to collect IOAM data, it adds IOAM TLV in the SRH. Based on the size of the segment list (SL), the ingress node preallocates space in the IOAM TLV.

If IOAM data from the last node in the segment-list (Egress node) is desired, the ingress uses an Ultimate Segment Pop (USP) SID advertised by the Egress node.

The ingress node MAY also insert the IOAM data about the local information in the IOAM TLV in the SRH at index 0 of the preallocated IOAM TLV.

4.2. Intermediate SR Segment Endpoint Node

The SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is a local SID. As part of the SR Header processing as described in [I-D.ietf-6man-segment-routing-header] and [I-D.ietf-spring-srv6-network-programming], the SR Segment Endpoint node performs the following IOAM operations.

If an intermediate SR segment endpoint node is not capable of processing IOAM TLV, it simply ignores it. I.e., it does not have to look or process SRH TLV.

If an intermediate SR segment endpoint node is capable of processing IOAM TLV and the local SID supports IOAM data recording, it checks if any SRH TLV is present in the packet using procedures defined in [I-D.ietf-6man-segment-routing-header]. If the node finds IOAM TLV in the SRH it finds the local index at which it is expected to record the IOAM data. The local index is found using the SRH.SL field. The node records the IOAM data at the desired preallocated space.

4.3. Egress Node

The Egress node is the last node in the segment-list of the SRH. When IOAM data from the Egress node is desired, a USP SID advertised by the Egress node is used by the Ingress node.

The processing of IOAM TLV at the Egress node is similar to the processing of IOAM TLV at the SR Segment Endpoint Node. The only difference is that the Egress node may telemeter the IOAM data to an external entity.

5. IANA Considerations

IANA is requested to allocate a mutable SRH TLV Type for IOAM TLV data fields under registry name "Segment Routing Header TLVs" requested by [I-D.6man-segment-routing-header].

| SRH TLV Type | Description | Reference |
|-----------------------|--------------------------|---------------|
| TBA1 Greater than 128 | TLV for IOAM Data Fields | This document |

6. Security Considerations

The security considerations of SRv6 are discussed in [I-D.spring-srv6-network-programming] and [I-D.6man-segment-routing-header], and the security considerations of IOAM in general are discussed in [I-D.ietf-ippm-ioam-data].

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain.

7. Acknowledgements

The authors would like to thank Shwetha Bhandari and Vengada Prasad Govindan for the discussions on IOAM.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.
- [I-D.spring-srv6-network-programming] Filsfils, C. et al. "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming, work in progress.
- [I-D.6man-segment-routing-header] Previdi, S., Filsfils, C. et al, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header, work in progress.
- [I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and Bernier, D., "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data, work in progress.
- [I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.

8.2. Informative References

- [I-D.6man-extension-header-insertion] D. Voyer, et al., "Insertion of IPv6 Segment Routing Headers in a Controlled Domain", draft-voyer-6man-extension-header-insertion, work in progress.

Internet-Draft

In-situ OAM SRv6 encapsulation

Authors' Addresses

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Frank Brockners
Cisco Systems, Inc.
Germany

Email: fbrockne@cisco.com

Nagendra Kumar Nainar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Carlos Pignataro
Cisco Systems, Inc.

Email: cpignata@cisco.com

Cheng Li
Huawei

Email: chenglil13@huawei.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Gaurav Dawra
LinkedIn

Email: gdawra.ietf@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2020

Z. Ali
C. Filsfils
P. Camarillo
Cisco Systems
D. Voyer
Bell Canada
November 3, 2019

Building blocks for Slicing in Segment Routing Network
draft-ali-spring-network-slicing-building-blocks-02.txt

Abstract

This document describes how to build network slice using the Segment Routing based technology. It explains how the existing building blocks specified for the Segment Routing can be used for this purpose.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | | |
|------|--|----|
| 1 | Introduction..... | 2 |
| 2 | Segment Routing Policy..... | 3 |
| 2.1 | Flex-Algorithm Based SR Policies | 4 |
| 2.2 | On-demand SR policy | 5 |
| 2.3 | Automatic Steering | 6 |
| 2.4 | Inter-domain Considerations | 6 |
| 3 | TI-LFA and Microloop Avoidance..... | 7 |
| 4 | SR VPN..... | 7 |
| 5 | Stateless Service Programming..... | 7 |
| 6 | Operations, Administration, and Maintenance (OAM)..... | 8 |
| 7 | QoS..... | 8 |
| 8 | Orchestration at the Controller..... | 9 |
| 9 | Illustration..... | 9 |
| 10 | Security Considerations | 9 |
| 11 | IANA Considerations | 10 |
| 12 | References | 10 |
| 12.1 | Normative References | 10 |
| 12.2 | | 10 |
| 13 | Acknowledgments | 10 |
| 14 | Contributors | 10 |

1 Introduction

As more and more Service Providers and Enterprises operate a single network infrastructure to support an ever-increasing number of services, the ability to custom fit transport to application needs is critically important. This includes creating network slices with different characteristics can coexist on top of the shared network infrastructure.

Network Slicing is meant to create (end-to-end) partitioned network infrastructure that can be used to provide differentiated connectivity behaviors to fulfill the requirements of a diverse set of services. Services belonging to different Network slices can be wholly disjoint or can share different parts of the network infrastructure. Network Slicing is one of the requirements in 5G [3GPP 23501].

Segment Routing enables Service Providers to support Network Slicing without any additional protocol, other than the SR IGP extensions. The network as a whole, in a distributed and entirely automated manner, can share a single infrastructure resource along multiple virtual services (slices). For example, one slice is optimized continuously for low-cost transport; a second Slice is optimized continuously for low-latency transport; a third Slice is orchestrated to support disjoint

services, etc. The optimization objective of each of these slices is programmable by the operator.

The Segment Routing specification already contains the various building blocks required to create network slices. This includes the following.

- . SR Policy with or without Flexible Algorithm.
- . TI-LFA with O(50 msec) protection in the slice underlay.
- . SR VPN.
- . SR Service Programming (NFV, SFC).
- . Operation, Administration and Management (OAM) and Performance Management (PM).
- . QoS using DiffServ.
- . Orchestration at the Controller.

Each of these building blocks works independently of each other. Their functionality can be combined to satisfy service provider's requirement for the Network Slicing. An external controller plays an important role to orchestrate these building blocks into a Slicing service.

This document elaborates on the attributes of each of these building blocks for Network Slicing. The document also highlights how services in each Slice can benefit from traffic engineering, network function virtualization/ service chaining (service programming), OAM, performance management, SDN readiness, O (50 msec) TI-LFA protection, etc. features of SR while respecting resource partitioning employed over the common networking infrastructure.

The document equally applicable to the MPLS and SRv6 instantiations of segment routing.

The following subsection elaborates on each of these build blocks.

2 Segment Routing Policy

Segment Routing (SR) allows a headend node to steer a packet flow along any path without creating intermediate per-flow states [I-D.ietf-spring-segment-routing-policy]. The headend node steers a flow into a Segment Routing Policy (SR Policy). I.e., the SR Policy can be used to steer traffic along any arbitrary path in the network. This allows operators to enforce low-latency and / or disjoint paths, regardless of the normal forwarding paths.

The SR policy is able to support various optimization objectives [I-D.draft-filsfils-spring-sr-policy-considerations]. The optimization objectives can be instantiated for the IGP metric ([RFC1195] [RFC2328] [RFC5340]) xor the TE metric ([RFC5305], [RFC3630]) xor the latency extended TE metric ([RFC7810] [RFC7471]). In addition, an SR policy is able to various constraints, including inclusion and/or exclusion of TE affinity, inclusion and/or exclusion of IP address, inclusion and/or exclusion of SRLG, inclusion and/or exclusion of admin-tag, maximum accumulated metric (IGP, TE, and latency), maximum number of SIDs in the solution SID-List, maximum number of weighted SID-Lists in the solution set, diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends), etc. [I-D.draft-filsfils-spring-sr-policy-considerations]. The supports for various optimization objectives and constraints enables SR policy to create Slices in the network.

SR policy can be instantiated with or without IGP Flexible Algorithm feature. The following subsection describes the SR Flexible Algorithm feature and how SR policy can utilize this feature.

2.1 Flex-Algorithm Based SR Policies

Flexible Algorithm enriches the SR Policy solution by adding additional segments having different properties than the IGP Prefix segments. Flex Algo adds flexible, user-defined segments to the SRTE toolbox. Specifically, it allows for association of the "intent" to Prefix SIDs. [I-D.ietf-lsr-flex-algo] defines the IGP based Flex-Algorithm solution which allows IGPs themselves to compute paths constraint by the "intent" represented by the Flex-Algorithm.

The Flex-Algorithm has the following attributes:

- . Algorithm associate to the SID a specific TE intent expressed as an optimization objective (an algorithm) [I-D.ietf-lsr-flex-algo].
- . Flexibility includes the ability of network operators to define the intent of each algorithm they implement.
- . By design the mapping between the Flex-Algorithm and its meaning is flexible and is defined by the user.
- . Flexibility also includes ability for operators to make the decision to exclude some specific links from the shortest path computation, e.g.,

- o operator 1 may define Algo 128 to compute the shortest path for TE metric and exclude red affinity links.
- o operator 2 may define Algo 128 to compute the shortest path for latency metric and exclude blue affinity links.

A Network Slice can be created by associating a Flexible-Algorithm value with the Slice via provisioning.

Flex Alg leverages SR on-demand next hop (ODN) and Automated Steering for intent-based instantiation of traffic engineered paths described in the following sub-sections. Specifically, as specified in [I-D.ietf-spring-segment-routing] the IGP Flex Algo Prefix SIDs can also be used as segments within SR Policies thereby leveraging the underlying IGP Flex Algo solution.

2.2 On-demand SR policy

Segment Routing On-Demand Next-hop (ODN) functionality enables on-demand creation of SR Policies for service traffic. Using a Path Computation Element (PCE), end-to-end SR Policy paths can be computed to provide end-to-end Segment Routing connectivity, even in multi-domain networks running with or without IGP Flexible-Algorithm [I-D.draft-ietf-spring-segment-routing-policy].

The On-Demand Next-hop functionality provides optimized service paths to meet customer and application SLAs (such as latency, disjointness) without any pre-configured TE tunnel and with the automatic steering of the service traffic on the SR Policy without a static route, autoroute-announce, or policy-based routing.

With this functionality, a Network Service Orchestrator can deploy the service based on their requirements. The service head-end router requests the PCE to compute the path for the service and then instantiates an SR Policy with the computed path and steers the service traffic into that SR Policy. If the topology changes, the stateful PCE updates the SR Policy path. This happens seamlessly, while TI-LFA protects the traffic in case the topology change happened due to a failure.

2.3 Automatic Steering

Automatically steering traffic into a Network Slice is one of the fundamental requirement for Slicing. That is made possible by the "Automated Steering" functionality of SR. Specifically, SR policy can be used for traffic engineer paths within a slice, "automatically steer" traffic to the right slice and connect IGP Flex-Algorithm domains sharing the same "intent".

A headend can steer a packet flow into a valid SR Policy within a slice in various ways [I-D.draft-ietf-spring-segment-routing-policy]:

- . Incoming packets have an active SID matching a local Binding SID (BSID) at the headend.
- . Per-destination Steering: incoming packets match a BGP/Service route which recurses on an SR policy.
- . Per-flow Steering: incoming packets match or recurse on a forwarding array of where some of the entries are SR Policies.
- . Policy-based Steering: incoming packets match a routing policy which directs them on an SR policy.

2.4 Inter-domain Considerations

The network slicing needs to be extended across multiple domains such that each domain can satisfy the intent consistently. SR has native inter-domain mechanisms, e.g., SR policies are designed to span multiple domains using a PCE based solution [I-D.ietf-spring-segment-routing], [I-D.ietf-spring-segment-routing-central-epe]. An edge router upon service configuration automatically requests to the Segment Routing PCE an inter-domain path to the remote service endpoint. The path can either be for simple best-effort inter-domain reachability or for reachability with an SLA contract and can be restricted to a Network Slice.

The SR native mechanisms for inter-domain are easily extendable to include the case when different IGP Flex-Algorithm values are used to represent the same intent. E.g., in domain1 Service Provider 1 (SP1) may use flex-algo 128 to indicate low latency Slice and in domain2 Service Provider 2 (SP2) may use flex-algo 129 to indicate low latency Slice. When an automation system at a PE1 in SP1 network configures a service with next hop (PE2) in SP2 network, SP1 contacts a Path Computation Element (PCE) to find a route to PE2. In the request, the PE1 also indicates the intent (i.e., the Flex-Algo 128) in the PCEP message. As the PCE has a complete understanding of both Domains, it can understand the path computation in Domain1 needs to be performed for Algorithm 128 and path computation in Domain2 needs to be

performed for Algorithm 129 (i.e., in the Low Latency Network Slice in both domains).

3 TI-LFA and Microloop Avoidance

The Segment Routing-based fast-reroute solution, TI-LFA, can provide per-destination sub-50msec protection upon any single link, node or SRLG failure regardless of the topology. The traffic is rerouted straight to the post-convergence path, hence avoiding any intermediate flap via an intermediate path. The primary and backup path computation is completely automatic by the IGP.

[I-D.draft-bashandy-rtgwg-segment-routing-ti-lfa] proposes a Topology Independent Loop-free Alternate Fast Re-route (TI-LFA), aimed at protecting node and adjacency segments within $O(50 \text{ msec})$ in the Segment Routing networks. Furthermore, [I-D. draft-bashandy-rtgwg-segment-routing-uloop] provides a mechanism leveraging Segment Routing to ensure loop-freeness during the IGP reconvergence process following a link-state change event.

As mentioned earlier, Network Slicing in Segment Routing works seamlessly with all the other components of the Segment Routing. This, of course, includes TI-LFA and microloop avoidance within a Slice, with the added benefit that backup path only uses resources available to the Slice. For example, when Flexible Algorithm is used, the TI-LFA backup path computation is performed such that it is optimized per Flexible-Algorithm. The backup path shares the same properties as the primary path. The backup path does not use a resource outside the Slice of the primary path it is protecting.

4 SR VPN

Virtual Private Networks (VPNs) provides a mean for creating a logically separated network to a different set of users access to a common network. Segment Routing is equipped with the rich multi-service virtual private network (VPN) capabilities, including Layer 3 VPN (L3VPN), Virtual Private Wire Service (VPWS), Virtual Private LAN Service (VPLS), and Ethernet VPN (EVPN). The ability of Segment Routing to support different VPN technologies is one of the fundamental building blocks for creating slicing an SR network.

5 Stateless Service Programming

An important part of the Network Slicing is the orchestration of virtualized service containers. [I-D.draft-xuclad-spring-sr-service-chaining] describes how to implement service segments and achieve stateless service programming in SR-MPLS and SRv6 networks. It introduces the notion of service segments. The ability of encoding the service segments along with the topological segment enables service providers to forward packets along a specific network path, but also steer them through VNFs or physical service appliances available in the network.

In an SR network, each of the service, running either on a physical appliance or in a virtual environment, is associated with a segment identifier (SID) for the service. These service SIDs are then leveraged as part of a SID-list to steer packets through the corresponding services. Service SIDs may be combined with topological SIDs to achieve service programming while steering the traffic through a specific topological path in the network. In this fashion, SR provides a fully integrated solution for overlay, underlay and service programming building blocks needed to satisfy network slicing requirements.

6 Operations, Administration, and Maintenance (OAM)

There are various OAM elements that are critical to satisfy Network Slicing requirements. These includes but not limited to the following:

- . Measuring per-link TE Matric.
- . Flooding per-link TE Matric.
- . Taking TE Matric into account during path calculation.
- . Taking TE Matric bound into account during path calculation.
- . SLA Monitoring: Service Provider can monitor each SR Policy in a Slice to Monitor SLA offered by the Policy using technique described in [I-D.draft-gandhi-spring-udp-pm]. This includes monitoring end-to-end delays on all ECMP paths of the Policy as well as monitoring traffic loss on a Policy. Remedial mechanisms can be used to ensure that the SR policy conforms to the SLA contract.

7 QoS

Segment Routing relies on MPLS and IP Differentiated Services. Differentiated services enhancements are intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. [RFC2475] defines

an architecture for implementing scalable service differentiation in the Internet. This architecture is composed of many functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing.

The DiffServ architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic depending on the traffic marker. Specifically, the node at the ingress of the DiffServ domain conditions, classifies and marks the traffic into a limited number of traffic classes. The function is used to ensure that the slice's traffic conforms to the contract associated with the slice.

Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams. Specifically, per class scheduling and queuing control mechanisms are applied at each IP hop to the traffic classes depending on packet's marking. Techniques such as queue management and a variety of scheduling mechanisms are used to get the required packet behavior to meet the slice's SLA.

8 Orchestration at the Controller

A controller plays a vital role in orchestrating the SR building blocks discussed above to create Network Slices. The controller also performs admission control and traffic placement for slice management at the transport layer. The SDN friendliness of the SR technology becomes handy to realize the orchestration. The controller may use PCEP or Netconf to interact with the routers. The router implements Yang model for SR-based network slicing.

Specification of the controller technology for orchestrating Network Slices, services and admission control for the services is outside the scope of this draft.

9 Illustration

To be added in a later revision.

10 Security Considerations

This document does not impose any additional security challenges.

11 IANA Considerations

This document does not define any new protocol or any extension to an existing protocol.

12 References

12.1 Normative References

7.2. Informative References

[I-D.filsfils-spring-segment-routing-policy]

Filsfils, C., Sivabalan, et al, "Segment Routing Policy For Traffic Engineering", draft-filsfils-spring-segment-routing-policy-05 (work in progress), February 2018.

[I-D.ietf-lsr-flex-algo] P. Psenak, et al,

draft-ietf-lsr-flex-algo, work in progress.

[I-D.ietf-spring-segment-routing]

Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.draft-filsfils-spring-sr-policy-considerations]

Filsfils, C., et al. draft-filsfils-spring-sr-policy-considerations (work in progress)

13 Acknowledgments

14 Contributors

Internet-Draft

Network Slicing Using SR

Authors' Addresses

Zafar Ali
Cisco Systems, Inc.
Email: zali@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Email: cf@cisco.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Email: pcamaril@cisco.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2021

Z. Ali
C. Filsfils
P. Camarillo
Cisco Systems
D. Voyer
Bell Canada
S. Matsushima
Softbank
February 21, 2021

Building blocks for Slicing in Segment Routing Network
draft-ali-spring-network-slicing-building-blocks-04.txt

Abstract

This document describes how to build network slice using the Segment Routing based technology. It explains how the building blocks specified for the Segment Routing can be used for this purpose.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 21, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | | |
|------|--|----|
| 1 | Introduction..... | 2 |
| 2 | Segment Routing Policy..... | 3 |
| 2.1 | Flex-Algorithm Based SR Policies | 4 |
| 2.2 | On-demand SR policy | 5 |
| 2.3 | Automatic Steering | 6 |
| 2.4 | Inter-domain Considerations | 6 |
| 3 | TI-LFA and Microloop Avoidance..... | 7 |
| 4 | SR VPN..... | 7 |
| 5 | Stateless Service Programming..... | 7 |
| 6 | Operations, Administration, and Maintenance (OAM)..... | 8 |
| 7 | QoS..... | 8 |
| 8 | Stateless Network Slice Identification..... | 9 |
| 8.1 | Stateless Slice Identification in SRv6..... | 9 |
| 8.2 | Stateless Slice Identification in SR-MPLS..... | 10 |
| 8 | Orchestration at the Controller..... | 10 |
| 9 | Illustration..... | 10 |
| 10 | Security Considerations | 10 |
| 11 | IANA Considerations | 11 |
| 12 | References | 11 |
| 12.1 | Normative References | 11 |
| 7.2 | | 11 |
| 13 | Acknowledgments | 11 |
| 14 | Contributors | 11 |

1 Introduction

As more and more Service Providers and Enterprises operate a single network infrastructure to support an ever-increasing number of services, the ability to custom fit transport to application needs is critically important. This includes creating network slices with different characteristics can coexist on top of the shared network infrastructure.

Network Slicing is meant to create (end-to-end) partitioned network infrastructure that can be used to provide differentiated connectivity behaviors to fulfill the requirements of a diverse set of services. Services belonging to different Network slices can be wholly disjoint or can share different parts of the network infrastructure. Network Slicing is one of the requirements in 5G [3GPP 23501].

Segment Routing enables Service Providers to support Network Slicing without any additional protocol, other than the SR IGP extensions. The network as a whole, in a distributed and entirely automated manner, can share a single infrastructure resource along multiple virtual services (slices). For example, one slice is optimized continuously for low-cost transport; a second Slice is optimized continuously for low-latency transport; a third Slice is orchestrated to support disjoint

services, etc. The optimization objective of each of these slices is programmable by the operator.

The Segment Routing specification already contains the various building blocks required to create network slices. This includes the following.

- . SR Policy with or without Flexible Algorithm.
- . TI-LFA with O(50 msec) protection in the slice underlay.
- . SR VPN.
- . SR Service Programming (NFV, SFC).
- . Operation, Administration and Management (OAM) and Performance Management (PM).
- . QoS using DiffServ.
- . Stateless Network Slice Identification
- . Orchestration at the Controller.

Each of these building blocks works independently of each other. Their functionality can be combined to satisfy service provider's requirement for the Network Slicing. An external controller plays an important role to orchestrate these building blocks into a Slicing service.

This document elaborates on the attributes of each of these building blocks for Network Slicing. The document also highlights how services in each Slice can benefit from traffic engineering, network function virtualization/ service chaining (service programming), OAM, performance management, SDN readiness, O (50 msec) TI-LFA protection, etc. features of SR while respecting resource partitioning employed over the common networking infrastructure.

The document equally applicable to the MPLS and SRv6 instantiations of segment routing.

The following subsection elaborates on each of these build blocks.

2 Segment Routing Policy

Segment Routing (SR) allows a headend node to steer a packet flow along any path without creating intermediate per-flow states [I-D.ietf-spring-segment-routing-policy]. The headend node steers a flow into a Segment Routing Policy (SR Policy). I.e., the SR Policy can be used to steer traffic along any arbitrary path in the network. This allows operators to enforce low-latency and / or disjoint paths, regardless of the normal forwarding paths.

The SR policy is able to support various optimization objectives [I-D.draft-filsfils-spring-sr-policy-considerations]. The optimization objectives can be instantiated for the IGP metric ([RFC1195] [RFC2328] [RFC5340]) xor the TE metric ([RFC5305], [RFC3630]) xor the latency extended TE metric ([RFC7810] [RFC7471]). In addition, an SR policy is able to various constraints, including inclusion and/or exclusion of TE affinity, inclusion and/or exclusion of IP address, inclusion and/or exclusion of SRLG, inclusion and/or exclusion of admin-tag, maximum accumulated metric (IGP, TE, and latency), maximum number of SIDs in the solution SID-List, maximum number of weighted SID-Lists in the solution set, diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends), etc. [I-D.draft-filsfils-spring-sr-policy-considerations]. The supports for various optimization objectives and constraints enables SR policy to create Slices in the network.

SR policy can be instantiated with or without IGP Flexible Algorithm feature. The following subsection describes the SR Flexible Algorithm feature and how SR policy can utilize this feature.

2.1 Flex-Algorithm

Flexible Algorithm enriches the SR Policy solution by adding additional segments having different properties than the IGP Prefix segments. Flex Algo adds flexible, user-defined segments to the SRTE toolbox. Specifically, it allows for association of the "intent" to Prefix SIDs. [I-D.ietf-lsr-flex-algo] defines the IGP based Flex-Algorithm solution which allows IGPs themselves to compute paths constraint by the "intent" represented by the Flex-Algorithm.

The Flex-Algorithm has the following attributes:

- . Algorithm associate to the SID a specific TE intent expressed as an optimization objective (an algorithm) [I-D.ietf-lsr-flex-algo].
- . Flexibility includes the ability of network operators to define the intent of each algorithm they implement.
- . By design the mapping between the Flex-Algorithm and its meaning is flexible and is defined by the user.
- . Flexibility also includes ability for operators to make the decision to exclude some specific links from the shortest path computation, e.g.,

- o operator 1 may define Algo 128 to compute the shortest path for TE metric and exclude red affinity links.
- o operator 2 may define Algo 128 to compute the shortest path for latency metric and exclude blue affinity links.

A Network Slice can be created by associating a Flexible-Algorithm value with the Slice via provisioning.

Flex Alg leverages SR on-demand next hop (ODN) and Automated Steering for intent-based instantiation of traffic engineered paths described in the following sub-sections. Specifically, as specified in [RFC8402] the IGP Flex Algo Prefix SIDs can also be used as segments within SR Policies thereby leveraging the underlying IGP Flex Algo solution.

2.2 On-demand SR policy

Segment Routing On-Demand Next-hop (ODN) functionality enables on-demand creation of SR Policies for service traffic. Using a Path Computation Element (PCE), end-to-end SR Policy paths can be computed to provide end-to-end Segment Routing connectivity, even in multi-domain networks running with or without IGP Flexible-Algorithm [I-D.draft-ietf-spring-segment-routing-policy].

The On-Demand Next-hop functionality provides optimized service paths to meet customer and application SLAs (such as latency, disjointness) without any pre-configured TE tunnel and with the automatic steering of the service traffic on the SR Policy without a static route, autoroute-announce, or policy-based routing.

With this functionality, a Network Service Orchestrator can deploy the service based on their requirements. The service head-end router requests the PCE to compute the path for the service and then instantiates an SR Policy with the computed path and steers the service traffic into that SR Policy. If the topology changes, the stateful PCE updates the SR Policy path. This happens seamlessly, while TI-LFA protects the traffic in case the topology change happened due to a failure.

2.3 Automatic Steering

Automatically steering traffic into a Network Slice is one of the fundamental requirement for Slicing. That is made possible by the "Automated Steering" functionality of SR. Specifically, SR policy can be used for traffic engineer paths within a slice, "automatically steer" traffic to the right slice and connect IGP Flex-Algorithm domains sharing the same "intent".

A headend can steer a packet flow into a valid SR Policy within a slice in various ways [I-D.draft-ietf-spring-segment-routing-policy]:

- . Incoming packets have an active SID matching a local Binding SID (BSID) at the headend.
- . Per-destination Steering: incoming packets match a BGP/Service route which recurses on an SR policy.
- . Per-flow Steering: incoming packets match or recurse on a forwarding array of where some of the entries are SR Policies.
- . Policy-based Steering: incoming packets match a routing policy which directs them on an SR policy.

2.4 Inter-domain Considerations

The network slicing needs to be extended across multiple domains such that each domain can satisfy the intent consistently. SR has native inter-domain mechanisms, e.g., SR policies are designed to span multiple domains using a PCE based solution [I-D.ietf-spring-segment-routing], [I-D.ietf-spring-segment-routing-central-epe]. An edge router upon service configuration automatically requests to the Segment Routing PCE an inter-domain path to the remote service endpoint. The path can either be for simple best-effort inter-domain reachability or for reachability with an SLA contract and can be restricted to a Network Slice.

The SR native mechanisms for inter-domain are easily extendable to include the case when different IGP Flex-Algorithm values are used to represent the same intent. E.g., in domain1 Service Provider 1 (SP1) may use flex-algo 128 to indicate low latency Slice and in domain2 Service Provider 2 (SP2) may use flex-algo 129 to indicate low latency Slice. When an automation system at a PE1 in SP1 network configures a service with next hop (PE2) in SP2 network, SP1 contacts a Path Computation Element (PCE) to find a route to PE2. In the request, the PE1 also indicates the intent (i.e., the Flex-Algo 128) in the PCEP message. As the PCE has a complete understanding of both Domains, it can understand the path computation in Domain1 needs to be performed for Algorithm 128 and path computation in Domain2 needs to be

performed for Algorithm 129 (i.e., in the Low Latency Network Slice in both domains).

3 TI-LFA and Microloop Avoidance

The Segment Routing-based fast-reroute solution, TI-LFA, can provide per-destination sub-50msec protection upon any single link, node or SRLG failure regardless of the topology. The traffic is rerouted straight to the post-convergence path, hence avoiding any intermediate flap via an intermediate path. The primary and backup path computation is completely automatic by the IGP.

[I-D.draft-bashandy-rtgwg-segment-routing-ti-lfa] proposes a Topology Independent Loop-free Alternate Fast Re-route (TI-LFA), aimed at protecting node and adjacency segments within $O(50 \text{ msec})$ in the Segment Routing networks. Furthermore, [I-D. draft-bashandy-rtgwg-segment-routing-uloop] provides a mechanism leveraging Segment Routing to ensure loop-freeness during the IGP reconvergence process following a link-state change event.

As mentioned earlier, Network Slicing in Segment Routing works seamlessly with all the other components of the Segment Routing. This, of course, includes TI-LFA and microloop avoidance within a Slice, with the added benefit that backup path only uses resources available to the Slice. For example, when Flexible Algorithm is used, the TI-LFA backup path computation is performed such that it is optimized per Flexible-Algorithm. The backup path shares the same properties as the primary path. The backup path does not use a resource outside the Slice of the primary path it is protecting.

4 SR VPN

Virtual Private Networks (VPNs) provides a mean for creating a logically separated network to a different set of users access to a common network. Segment Routing is equipped with the rich multi-service virtual private network (VPN) capabilities, including Layer 3 VPN (L3VPN), Virtual Private Wire Service (VPWS), Virtual Private LAN Service (VPLS), and Ethernet VPN (EVPN). The ability of Segment Routing to support different VPN technologies is one of the fundamental building blocks for creating slicing an SR network.

5 Stateless Service Programming

An important part of the Network Slicing is the orchestration of virtualized service containers. [I-D.draft-xuclad-spring-sr-service-chaining] describes how to implement service segments and achieve stateless service programming in SR-MPLS and SRv6 networks. It introduces the notion of service segments. The ability of encoding the service segments along with the topological segment enables service providers to forward packets along a specific network path, but also steer them through VNFs or physical service appliances available in the network.

In an SR network, each of the service, running either on a physical appliance or in a virtual environment, is associated with a segment identifier (SID) for the service. These service SIDs are then leveraged as part of a SID-list to steer packets through the corresponding services. Service SIDs may be combined with topological SIDs to achieve service programming while steering the traffic through a specific topological path in the network. In this fashion, SR provides a fully integrated solution for overlay, underlay and service programming building blocks needed to satisfy network slicing requirements.

6 Operations, Administration, and Maintenance (OAM)

There are various OAM elements that are critical to satisfy Network Slicing requirements. These includes but not limited to the following:

- . Measuring per-link TE Matric.
- . Flooding per-link TE Matric.
- . Taking TE Matric into account during path calculation.
- . Taking TE Matric bound into account during path calculation.
- . SLA Monitoring: Service Provider can monitor each SR Policy in a Slice to Monitor SLA offered by the Policy using technique described in [I-D.draft-gandhi-spring-udp-pm]. This includes monitoring end-to-end delays on all ECMP paths of the Policy as well as monitoring traffic loss on a Policy. Remedial mechanisms can be used to ensure that the SR policy conforms to the SLA contract.

7 QoS

Segment Routing relies on MPLS and IP Differentiated Services. Differentiated services enhancements are intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. [RFC2475] defines

an architecture for implementing scalable service differentiation in the Internet. This architecture is composed of many functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing.

The DiffServ architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic depending on the traffic marker. Specifically, the node at the ingress of the DiffServ domain conditions, classifies and marks the traffic into a limited number of traffic classes. The function is used to ensure that the slice's traffic conforms to the contract associated with the slice.

Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams. Specifically, per class scheduling and queuing control mechanisms are applied at each IP hop to the traffic classes depending on packet's marking. Techniques such as queue management and a variety of scheduling mechanisms are used to get the required packet behavior to meet the slice's SLA.

8 Stateless Network Slice Identification

Some use-cases require a slice identifier (SLID) in the packet to provide differentiated treatment of the packets belonging to different network slices.

The network slice instantiation using the SLID in the packet is required to work with the building blocks described in the previous sections. For example, the QoS/ DiffServ needs to be observed on a per slice basis. The slice identification needs to be topologically independent and stateless.

8.1 Stateless Slice Identification in SRv6

[I-D.draft-filsfils-spring-srv6-stateless-slice-id] describes a stateless encoding of slice identification in the outer IPv6 header of an SRv6 domain. As defined in RFC8754 [RFC8754], when an ingress PE receives a packet that traverses the SR domain, it encapsulates the packet in an outer IPv6 header and an optional SRH. Based on a local policy of the SR domain, the Flow Label field of the outer IPv6 header carries the SLID. Specifically, the SLID is added in the 8 most significant bits of the Flow Label field of the outer IPv6 header. The remaining 12 bits of the Flow Label field are set as described in section 5.5 of [RFC8754] for inter-domain packets. Based on the local policy of the SR domain, the draft also uses one of the bits in the Traffic Class field of the outer IPv6 header to indicate that the entropy label contains the SLID.

The network slicing mechanism described in [I-D.draft-filsfils-spring-srv6-stateless-slice-id] works seamlessly with the building blocks described in the previous sections. For example, the slice identification is independent of topology and the network's QoS/DiffServ policy. It enables scalable network slicing for SRv6 overlays.

8.2 Stateless Slice Identification in SR-MPLS

[I-D.draft-decraene-mpls-slid-encoded-entropy-label-id] describes a similar stateless encoding of slice identification in the SR-MPLS domain. Specifically, the document extends the use of the Entropy Label to carry the SLID. The number of bits to be used for encoding the SLID in the Entropy Label is governed by a local policy of the SR domain. Based on the local policy of the SR domain, the draft uses one of the bits in the TTL field of the Entropy Label to indicate that the Entropy Label contains the SLID.

The network slicing mechanism described in [I-D.draft-decraene-mpls-slid-encoded-entropy-label-id] works seamlessly with the building blocks described in the previous sections. For example, the slice identification is independent of topology and the network's QoS/DiffServ policy. It enables scalable network slicing for SR-MPLS overlays.

8 Orchestration at the Controller

A controller plays a vital role in orchestrating the SR building blocks discussed above to create Network Slices. The controller also performs admission control and traffic placement for slice management at the transport layer. The SDN friendliness of the SR technology becomes handy to realize the orchestration. The controller may use PCEP or Netconf to interact with the routers. The router implements Yang model for SR-based network slicing.

Specification of the controller technology for orchestrating Network Slices, services and admission control for the services is outside the scope of this draft.

9 Illustration

To be added in a later revision.

10 Security Considerations

This document does not impose any additional security challenges.

11 IANA Considerations

This document does not define any new protocol or any extension to an existing protocol.

12 References

12.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

7.2. Informative References

- [I-D.ietf-spring-segment-routing-policy] Filsfils, C., Sivabalan, et al, "Segment Routing Policy For Traffic Engineering", draft-ietf-spring-segment-routing-policy (work in progress).
- [I-D.ietf-lsr-flex-algo] P. Psenak, et al, draft-ietf-lsr-flex-algo, work in progress.
- [RFC8402] Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC8402.
- [I-D.draft-filsfils-spring-sr-policy-considerations] Filsfils, C., et al. draft-filsfils-spring-sr-policy-considerations (work in progress)
- [RFC8754] Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-16 (work in progress), February 2019.
- [I-D.draft-filsfils-spring-srv6-stateless-slice-id] Filsfils, C., et al. draft-filsfils-spring-srv6-stateless-slice-id, work in progress.
- [I-D.draft-decraene-mpls-slid-encoded-entropy-label-id] Decraene B., Filsfils, C., Henderickx W., Saad T., Beeram V., work in progress.

13 Acknowledgments

14 Contributors

Francois Clad
Cisco Systems, Inc.
fclad@cisco.com

Internet-Draft

Network Slicing Using SR

Authors' Addresses

Zafar Ali
Cisco Systems, Inc.
Email: zali@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Email: cf@cisco.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Email: pcamaril@cisco.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2020

R. Bonica
S. Hegde
Juniper Networks
Y. Kamite
NTT Communications Corporation
A. Alston
D. Henriques
Liquid Telecom
L. Jalil
Verizon
J. Halpern
Ericsson
J. Linkova
Google
G. Chen
Baidu
October 15, 2019

Segment Routing Mapped To IPv6 (SRm6)
draft-bonica-spring-srv6-plus-06

Abstract

This document describes Segment Routing mapped to IPv6 (SRm6). SRm6 is a Segment Routing (SR) solution that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRm6 is optimized for ASIC-based forwarding devices that operate at high data rates.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 17, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Overview | 3 |
| 2. Requirements Language | 4 |
| 3. Paths, Segments And Instructions | 5 |
| 4. Segment Types | 6 |
| 4.1. Adjacency Segments | 6 |
| 4.2. Node Segments | 7 |
| 4.3. Binding Segments | 7 |
| 5. Segment Identifiers (SID) | 8 |
| 5.1. Range | 9 |
| 5.2. Assigning SIDs to Adjacency Segments | 10 |
| 5.3. Assigning SIDs to Node Segments | 11 |
| 5.4. Assigning SIDs to Binding Segments | 11 |
| 6. Service Instructions | 11 |
| 6.1. Per-Segment | 11 |
| 6.2. Per-Path | 12 |
| 7. The IPv6 Data Plane | 12 |
| 7.1. The Routing Header | 13 |
| 7.2. The Destination Options Header | 14 |
| 8. Control Plane | 15 |
| 9. Differences Between SRv6 and SRv6+ | 15 |
| 9.1. Routing Header Size | 15 |
| 9.2. Decoupling of Topological and Service Instructions | 17 |
| 9.3. Authentication | 17 |
| 9.4. Traffic Engineering Capability | 18 |
| 9.5. IP Addressing Architecture | 18 |
| 10. Compliance | 18 |
| 11. Operational Considerations | 19 |
| 11.1. Ping and Traceroute | 19 |
| 11.2. ICMPv6 Rate Limiting | 19 |
| 12. IANA Considerations | 19 |
| 13. Security Considerations | 19 |

| | |
|--|----|
| 14. Acknowledgements | 19 |
| 15. References | 19 |
| 15.1. Normative References | 19 |
| 15.2. Informative References | 21 |
| Authors' Addresses | 22 |

1. Overview

Network operators deploy Segment Routing (SR) [RFC8402] so that they can forward packets through SR paths. An SR path provides unidirectional connectivity from its ingress node to its egress node. While an SR path can follow the least cost path from ingress to egress, it can also follow any other path.

An SR path contains one or more segments. A segment provides unidirectional connectivity from its ingress node to its egress node. It includes a topological instruction that controls its behavior.

The topological instruction is executed on the segment ingress node. It determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node.

Likewise, a per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node. Section 3 of this document illustrates the relationship between SR paths, segments and instructions.

A Segment Identifier (SID) identifies each segment. Because there is a one-to-one relationship between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

An SR path can be represented by its ingress node as an ordered sequence of SIDs. In order to forward a packet through an SR path, the SR ingress node encodes the SR path into the packet as an ordered sequence of SIDs. It can also augment the packet with service instructions.

Because the SR ingress node is also the first segment ingress node, it executes the topological instruction associated with the first segment. This causes the packet to be forwarded to the first segment egress node. When the first segment egress node receives the packet, it executes any per-segment service instructions that augment the first segment.

If the SR path contains exactly one segment, the first segment egress node is also the path egress node. In this case, that node executes any per-path service instruction that augments the path, and SR forwarding is complete.

If the SR path contains multiple segments, the first segment egress node is also the second segment ingress node. In this case, that node executes the topological instruction associated with the second segment. The above-described procedure continues until the packet arrives at the SR egress node.

In the above-described procedure, only the SR ingress node maintains path information. Segment ingress and egress nodes maintain information regarding the segments in which they participate, but they do not maintain path information.

The SR architecture, described above, can leverage either an MPLS [RFC3031] data plane or an IPv6 [RFC8200] data plane. SR-MPLS [I-D.ietf-spring-segment-routing-mpls] leverages MPLS. SRv6 [I-D.ietf-spring-srv6-network-programming] [I-D.ietf-6man-segment-routing-header] leverages IPv6.

This document describes Segment Routing mapped to IPv6 (SRm6). SRm6 is an SR variant that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRm6 is optimized for ASIC-based forwarding devices that operate at high data rates. Section 9 of this document highlights differences between SRv6 and SRm6.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Paths, Segments And Instructions

An SRm6 path is determined by the segments that it contains. It can be represented by its ingress node as an ordered sequence of SIDs.

A segment is determined by its ingress node and by the topological instruction that controls its behavior. The topological instruction determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions augment, but do not determine, segments. A segment ingress node can:

- o Send one packet through a segment with one per-segment service instruction.
- o Send another packet through the same segment with a different per-segment service instruction.
- o Send another packet through the same segment without any per-segment service instructions.

Likewise, per-path service instructions augment, but do not determine, paths.

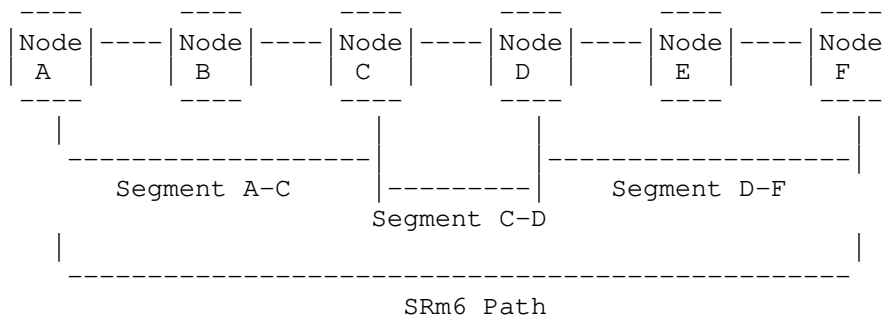


Figure 1: Paths, Segments And Instructions

Figure 1 depicts an SRm6 path. The path provides unidirectional connectivity from its ingress node (i.e., Node A) to its egress node (i.e., Node F). It contains Segment A-C, Segment C-D and Segment D-F.

In Segment A-C, Node A is the ingress node, Node B is a transit node, and Node C is the egress node. Therefore, the topological instruction that controls the segment is executed on Node A, while

per-segment service instructions that augment the segment (if any exist) are executed on Node C.

In Segment C-D, Node C is the ingress node and Node D is the egress node. Therefore, the topological instruction that controls the segment is executed on Node C, while per-segment service instructions that augment the segment (if any exist) are executed on Node D.

In Segment D-F, Node D is the ingress node, Node E is a transit node, and Node F is the egress node. Therefore, the topological instruction that controls the segment is executed on Node D, while per-segment service instructions that augment the segment (if any exist) are executed on Node F.

Node F is also the path egress node. Therefore, if a per-path service instruction augments the path, it is executed on Node F.

Segments A-C, C-D and D-F are also contained by other paths that are not included in the figure.

4. Segment Types

SRm6 supports the following segment types:

- o Adjacency.
- o Node.
- o Binding.

Adjacency segments forward packets through a specified link that connects the segment ingress node to the segment egress node. Node segments forward packets through the least cost path from the segment ingress node to the segment egress node. Binding segments facilitate recursive application of SRm6. They cause SRm6 paths to be nested in a hierarchy.

Each segment type is described below.

4.1. Adjacency Segments

When a packet is submitted to an adjacency segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and receives the following parameters:

- o An IPv6 address that identifies an interface on the segment egress node.

- o An interface identifier.

The topological instruction behaves as follows:

- o If the interface that was received as a parameter is not operational, discard the packet and send an ICMPv6 [RFC4443] Destination Unreachable message (Code: 5, Source Route Failed) to the packet's source node.
- o Overwrite the packet's Destination Address with the IPv6 address that was received as a parameter.
- o Forward the packet through the above-mentioned interface.

For further processing details, see [I-D.bonica-6man-comp-rtg-hdr].

4.2. Node Segments

When a packet is submitted to a node segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and receives an IPv6 address as a parameter. The IPv6 address identifies an interface on the segment egress node.

The topological instruction behaves as follows:

- o If the segment ingress node does not have a viable route to the IPv6 address received as a parameter, discard the packet and send an ICMPv6 Destination Unreachable message (Code:1 Net Unreachable) to the packet's source node.
- o Overwrite the packet's Destination Address with the destination address that was received as a parameter.
- o Forward the packet to the next hop along the least cost path to the segment egress node. If there are multiple least cost paths to the segment egress node (i.e., Equal Cost Multipath), execute procedures so that all packets belonging to a flow are forwarded through the same next hop.

For further processing details, see [I-D.bonica-6man-comp-rtg-hdr].

4.3. Binding Segments

When a packet is submitted to a binding segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and receives the following parameters:

- o An IPv6 address.
- o A SID list length.
- o A SID list.

The topological instruction behaves as follows:

- o If the segment ingress node does not have a viable route to the IPv6 address received as a parameter, discard the packet and send an ICMPv6 Destination Unreachable message (Code:1 Net Unreachable) to the packet's source node.
- o Prepend a Compressed Routing Header (CRH) [I-D.bonica-6man-comp-rtg-hdr] to the packet. Copy the SID list length, received as a parameter, to the CRH Segments Left field. Also copy the SID list, received as a parameter, to the CRH SID list.
- o Prepend an IPv6 header to the packet. Copy the IPv6 address, received as a parameter, to the IPv6 Destination Address.
- o Forward the packet to the next hop along the least cost path to the IPv6 address received as a parameter. If there are multiple least cost paths to the IPv6 address received as a parameter (i.e., Equal Cost Multipath), execute procedures so that all packets belonging to a flow are forwarded through the same next hop.

For further processing details, see [I-D.bonica-6man-comp-rtg-hdr].

5. Segment Identifiers (SID)

A Segment Identifier (SID) is an unsigned integer that identifies a segment. Because there is a one-to-one relationship between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

SIDs have node-local significance. This means that a segment ingress node MUST identify each segment that it originates with a unique SID.

However, a SID that is used by one segment ingress node to identify a segment that it originates can be used by another segment ingress node to identify another segment. For example, SID S can identify both of the following:

- o A segment whose ingress is Node A and whose egress is Node Z.
- o Another segment whose ingress is Node B and whose egress is also node Z.

Although SIDs have node-local significance, an SRm6 path can be uniquely identified by its ingress node and an ordered sequence of SIDs. This is because the topological instruction associated with each segment determines the ingress node of the next segment (i.e., the node upon which the next SID has significance.)

SIDs can be assigned in a manner that simplifies network operations. See Section 5.2 and Section 5.3 for details.

5.1. Range

SID values range from 0 to a configurable Maximum SID Value (MSV). The values 0 through 15 are reserved for future use. The following are valid MSVs:

- o 65,535 (i.e., $2^{16} - 1$).
- o 4,294,967,295 (i.e., $2^{32} - 1$).

In order to optimize packet encoding (Section 7.1), network operators can configure all nodes within an SRm6 domain to have the smallest feasible MSV. The following paragraphs explain how an operator determines the smallest feasible MSV.

Consider an SRm6 domain that contains 5,000 nodes connected to one another by point-to-point infrastructure links. The network topology is not a full-mesh. In fact, each node supports 200 point-to-point infrastructure links or fewer. Given this SRm6 domain, we will determine the smallest feasible MSV under the following conditions:

- o The SRm6 domain contains adjacency segments only.
- o The SRm6 domain contains node segments only.
- o The SRm6 domain contains both adjacency and node segments.

If an SRm6 domain contains adjacency segments only, and each node creates a adjacency segment to each of its neighbors, each node will

create 200 segments or fewer and consume 200 SIDs or fewer. This is because each node has 200 neighbors or fewer. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRm6 domain will not increase the smallest feasible MSV, so long as each node continues to support 65,519 point-to-point infrastructure links or fewer. If a single node is added to the domain and that node supports 65,520 infrastructure links, the smallest feasible MSV will increase to 4,294,967,295.

If an SRm6 domain contains node segments only, and every node creates a node segment to every other node, every node will create 4,999 segments and consume 4,999 SIDs. This is because the domain contains 5,000 nodes. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRm6 domain will not increase the smallest feasible MSV until the number of nodes exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

If an SRm6 domain contains both adjacency and node segments, each node will create 5,199 segments or fewer and consume 5,199 SIDs or fewer. This value is the sum of the following:

- o The number of node segments that each node will create, given that every node creates a node segment to every other node (i.e., 4,999).
- o The number of adjacency segments that each node will create, given that each node creates a adjacency segment to each of its neighbors (i.e., 200 or fewer).

Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRm6 domain will not increase the smallest feasible MSV until the number of nodes plus the maximum number of infrastructure links per node exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

5.2. Assigning SIDs to Adjacency Segments

Network operators can establish conventions by which they assign SIDs to adjacency segments. These conventions can simplify network operations.

For example, a network operator can reserved a range of SIDs for adjacency segments. It can further divide that range into subranges, so that all segments sharing a common egress node are identified by SIDs from the same subrange.

5.3. Assigning SIDs to Node Segments

In order to simplify network operations, all node segments that share a common egress node are identified by the same SID. In order to maintain this discipline, network wide co-ordination is required.

For example, assume that an SRm6 domain contains N nodes. Network administrators reserve a block of N SIDs and configure one of those SIDs on each node. Each node advertises its SID into the control plane. When another node receives that advertisement, it creates a node segment between itself and the advertising node. It also associates the SID that it received in the advertisement with the newly created segment. See [I-D.bonica-lsr-crh-isis-extensions] for details.

5.4. Assigning SIDs to Binding Segments

Network operators can establish conventions by which they assign SIDs to binding segments. These conventions can simplify network operations.

For example, a network operator can reserve a range of SIDs for binding segments. It can further divide that range into subranges, so that all segments sharing a common egress node are identified by SIDs from the same subrange.

6. Service Instructions

SRm6 supports the following service instruction types:

- o Per-segment.
- o Per-path.

Each is described below.

6.1. Per-Segment

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node. Because the path egress node is also a segment egress node, it can execute per-segment service instructions.

The following are examples of per-segment service instructions:

- o Expose a packet to a firewall policy.
- o Expose a packet to a sampling policy.

Per-segment Service Instruction Identifiers identify a set of service instructions. Per-segment Service Instruction Identifiers are allocated and distributed by a controller. They have domain-wide significance.

6.2. Per-Path

A per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node.

The following are examples of per-path service instructions:

- o De-encapsulate a packet and forward its newly exposed payload through a specified interface.
- o De-encapsulate a packet and forward its newly exposed payload using a specified routing table.

Per-path Service Instruction Identifiers identify per-path service instructions. Per-path Service Instruction Identifiers are allocated and distributed by the processing node (i.e., the path egress node). They have node-local significance. This means that the path egress node MUST allocate a unique Per-path Service Instruction Identifier for each per-path service instruction that it instantiates.

7. The IPv6 Data Plane

SRm6 ingress nodes generate IPv6 header chains that represent SRm6 paths. An IPv6 header chain contains an IPv6 header. It can also contain one or more extension headers.

An extension header chain that represents an SRm6 path can contain any valid combination of IPv6 extension headers. The following bullet points describe how SRm6 leverages IPv6 extension headers:

- o If an SRm6 path contains multiple segments, the IPv6 header chain that represents it MUST contain a Routing header. The SRm6 path MUST be encoded in the Routing header as an ordered sequence of SIDs.
- o If an SRm6 path is augmented by a per-path service instruction, the IPv6 header chain that represents it MUST contain a

Destination Options header. The Destination Options header MUST immediately precede an upper-layer header and it MUST include a Per-Path Service Instruction Identifier.

- o If an SRm6 path contains a segment that is augmented by a per-segment service instruction, the IPv6 chain that represents it MUST contain a Routing header and a Destination Options header. The Destination Options header MUST immediately precede a Routing header and it MUST include the Per-Segment Service Instruction Identifier.

The following subsections describe how SRm6 uses the Routing header and the Destination Options header.

7.1. The Routing Header

SRm6 defines two new Routing header types. Generically, they are called the Compressed Routing Header (CRH) [I-D.bonica-6man-comp-rtg-hdr]. More specifically, the 16-bit version of the CRH is called the CRH-16, while the 32-bit version of the CRH is called the CRH-32.

Both CRH versions contain the following fields:

- o Next Header - Identifies the header immediately following the CRH.
- o Hdr Ext Len - Length of the CRH.
- o Routing Type - Identifies the Routing header variant (i.e., CRH-16 or CRH-32).
- o Segments Left - The number of segments still to be traversed before reaching the path egress node.
- o SID List - Represents the SRm6 path as an ordered list of SIDs. SIDs are listed in reverse order, with SID[0] representing the final segment, SID[1] representing the penultimate segment, and so forth. SIDs are listed in reverse order so that Segments Left can be used as an index to the SID List. The SID indexed by Segments Left is called the current SID.

In the CRH-16, each SID list entry is encoded in 16-bits. In the CRH-32, each SID list entry is encoded in 32-bits. In networks where the smallest feasible MSV (Section 5.1) is greater than 65,635, CRH-32 is required. Otherwise, CRH-16 is preferred.

As per [RFC8200], when an IPv6 node receives a packet, it examines the packet's destination address. If the destination address

represents an interface belonging to the node, the node processes the next header. If the node encounters and recognizes the CRH, it processes the CRH as follows:

- o If Segments Left equal 0, skip over the CRH and process the next header in the packet.
- o Decrement Segments Left.
- o Search for the current SID in a local table that maps SID's to topological instructions. If the current SID cannot be found in that table, send an ICMPv6 Parameter Problem message to the packet's Source Address and discard the packet.
- o Execute the topological instruction found in the table as described in Section 4. This causes the packet to be forwarded to the segment egress node.

When the packet arrives at the segment egress node, the above-described procedure is repeated. For further processing details, see [I-D.bonica-6man-comp-rtg-hdr].

7.2. The Destination Options Header

According to [RFC8200], the Destination Options header contains one or more IPv6 options. It can occur twice within a packet, once before a Routing header and once before an upper-layer header. The Destination Options header that occurs before a Routing header is processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations that are listed in the Routing header. The Destination Options header that occurs before an upper-layer header is processed by the packet's final destination only.

Therefore, SRm6 defines the following new IPv6 options:

- o The SRm6 Per-Segment Service Instruction Option
[I-D.bonica-6man-seg-end-opt]
- o The SRm6 Per-Path Service Instruction Option
[I-D.bonica-6man-vpn-dest-opt]

The SRm6 Per-Segment Service Instruction Option is encoded in a Destination Options header that precedes the CRH. Therefore, it is processed by every segment egress node. It includes a Per-Segment Service Instruction Identifier and causes segment egress nodes to execute per-segment service instructions.

The SRm6 Per-Path Service Instruction Option is encoded in a Destination Options header that precedes the upper-layer header. Therefore, it is processed by the path egress node only. It includes a Per-Path Service Instruction Identifier and causes the path egress node to execute a per-path service instruction.

8. Control Plane

IS-IS extensions [I-D.bonica-lsr-crh-isis-extensions] have been defined for the following purposes:

- o So that SRm6 segment ingress nodes can flood information regarding adjacency segments that they originate.
- o So that SRm6 segment egress nodes can flood information regarding node segments that they terminate.

BGP extensions [I-D.ssangli-idr-bgp-vpn-srv6-plus] are defined so that SRm6 path egress nodes can associate path-terminating service instructions with Network Layer Reachability Information (NLRI). Additional BGP extensions [I-D.alston-spring-crh-bgp-signalling] are defined so that SIDs can be mapped to the IPv6 addresses that they represent.

9. Differences Between SRv6 and SRv6+

9.1. Routing Header Size

SRv6 defines a Routing header type, called the Segment Routing Header (SRH). The SRH contains a field that represents the SRv6 path as an ordered sequence of SIDs. Each SID contained by that field is 128 bits long.

Likewise, SRm6 defines two Routing Header Types, called CRH-16 and CRH-32. Both contain a field that represents the SRv6 path as an ordered sequence of SIDs. In the CRH-16, each SID is 16 bits long. In the CRH-32, each SID is 32 bits long.

| SIDs | SRv6 SRH (128-bit SID) | SRm6 CRH-16 | SRm6 CRH-32 |
|------|------------------------|-------------|-------------|
| 1 | 24 | 8 | 8 |
| 2 | 40 | 8 | 16 |
| 3 | 56 | 16 | 16 |
| 4 | 72 | 16 | 24 |
| 5 | 88 | 16 | 24 |
| 6 | 104 | 16 | 32 |
| 7 | 120 | 24 | 32 |
| 8 | 136 | 24 | 40 |
| 9 | 152 | 24 | 40 |
| 10 | 168 | 24 | N/A |
| 11 | 184 | 32 | N/A |
| 12 | 200 | 32 | N/A |
| 13 | 216 | 32 | N/A |
| 14 | 232 | 32 | N/A |
| 15 | 248 | 40 | N/A |
| 16 | 264 | 40 | N/A |
| 17 | 280 | 40 | N/A |
| 18 | 296 | 40 | N/A |

Table 1: Routing Header Size (in Bytes) As A Function Of Routing Header Type and Number Of SIDs

Table 1 reflects Routing header size as a function of Routing header type and number of SIDs contained by the Routing header. Due to their relative immaturity, [I-D.filsfils-spring-net-pgm-extension-srv6-usid], [I-D.li-spring-compressed-srv6-np] and [I-D.mirsky-6man-unified-id-sr] are omitted from this analysis.

Large Routing headers are undesirable for the following reasons:

- o Many ASIC-based forwarders copy the entire IPv6 extension header chain from buffer memory to on-chip memory. As the size of the IPv6 extension header chain increases, so does the cost of this copy.
- o Because Path MTU Discovery (PMTUD) [RFC8201] is not entirely reliable, many IPv6 hosts refrain from sending packets larger than the IPv6 minimum link MTU (i.e., 1280 bytes). When packets are small, the overhead imposed by large Routing headers becomes pronounced.

9.2. Decoupling of Topological and Service Instructions

SRm6 decouples topological instructions from service instructions. Topological instructions are invoked at the segment ingress node, as a result of CRH processing, while service instructions are invoked at the segment egress node, as a result of Destination Option processing. Therefore, network operators can use SRm6 mechanisms to support topological instructions, service instructions, or both.

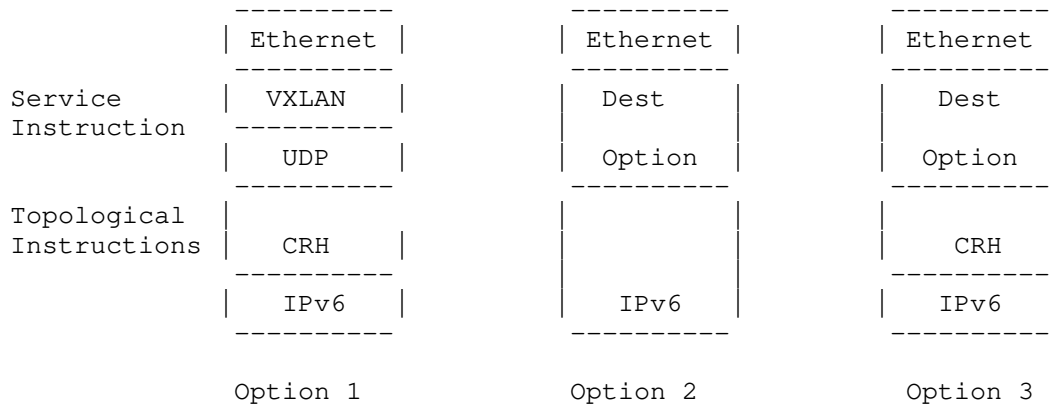


Figure 2: EVPN Design Alternatives

Figure 2 illustrates this point by depicting design options available to network operators offering Ethernet Virtual Private Network [RFC7432] services over Virtual eXtensible Local Area Network (VXLAN) [RFC7348]. In Option 1, the network operator encodes topological instructions in the CRH, while encoding service instructions in a VXLAN header. In Option 2, the network operator encodes service instructions in a Destination Options header, while allowing traffic to traverse the least cost path between the ingress and egress Provider Edge (PE) routers. In Option 3, the network operator encodes topological instructions in the CRH, and encodes service instructions in a Destination Options header.

9.3. Authentication

The IPv6 Authentication Header (AH) [RFC4302] can be used to authenticate SRm6 packets. However, AH processing is not defined in SRv6.

9.4. Traffic Engineering Capability

SRm6 supports traffic engineering solutions that rely exclusively upon adjacency segments. For example, consider an SRm6 network whose diameter is 12 hops and whose minimum feasible MSV is 65,525. In that network, in the worst case, SRm6 overhead is 72 bytes (i.e., a 40-byte IPv6 header and a 32-byte CRH-16).

SRv6 also supports traffic engineering solutions that rely exclusively upon adjacency segments (i.e., END.X SIDs). However, SRv6 overhead may be prohibitive. For example, consider an SRv6 network whose diameter is 12 hops. In the worst case, SRv6 overhead is 240 bytes (i.e., a 40 byte IPv6 header and a 200-byte SRH).

9.5. IP Addressing Architecture

In SRv6, an IPv6 address can represent either of the following:

- o A network interface
- o An instruction instantiated on a node (i.e., an SRv6 SID)

In SRm6 an IPv6 address always represents a network interface, as per [RFC4291].

10. Compliance

In order to be compliant with this specification, an SRm6 implementation MUST:

- o Be able to process IPv6 options as described in Section 4.2 of [RFC8200].
- o Be able to process the Routing header as described in Section 4.4 of [RFC8200].
- o Be able to process the Destination Options header as described in Section 4.6 of [RFC8200].
- o Support the CRH-16 and the CRH-32

Additionally, an SRm6 implementation MAY:

- o Recognize the Per-Segment Service Instruction Option.
- o Recognize the Per-Path Service Instruction Option.

11. Operational Considerations

11.1. Ping and Traceroute

Ping and Traceroute [RFC2151] both operate correctly in SRm6 (i.e., in the presence of the CRH).

11.2. ICMPv6 Rate Limiting

As per [RFC4443], SRm6 nodes rate limit the ICMPv6 messages that they emit.

12. IANA Considerations

SID values 0-15 are reserved for future use. They may be assigned by IANA, based on IETF Consensus.

IANA is requested to establish a "Registry of SRm6 Reserved SIDs". Values 0-15 are reserved for future use.

13. Security Considerations

SRm6 domains MUST NOT span security domains. In order to enforce this requirement, security domain edge routers MUST do one of the following:

- o Discard all inbound SRm6 packets whose IPv6 destination address represents domain infrastructure.
- o Authenticate [RFC4302] [RFC4303] all inbound SRm6 packets whose IPv6 destination address represents domain infrastructure.

14. Acknowledgements

The authors wish to acknowledge Dr. Vanessa Ameen, Reji Thomas, Parag Kaneriy, Rejesh Shetty, Nancy Shaw, and John Scudder.

15. References

15.1. Normative References

- [I-D.alston-spring-crh-bgp-signalling]
Alston, A., Henriques, D., and R. Bonica, "BGP Extensions for IPv6 Compressed Routing Header (CRH)", draft-alston-spring-crh-bgp-signalling-01 (work in progress), July 2019.

- [I-D.bonica-6man-comp-rtg-hdr]
Bonica, R., Kamite, Y., Niwa, T., Alston, A., Henriques, D., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Compressed Routing Header (CRH)", draft-bonica-6man-comp-rtg-hdr-07 (work in progress), September 2019.
- [I-D.bonica-6man-seg-end-opt]
Bonica, R., Halpern, J., Kamite, Y., Niwa, T., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The Per-Segment Service Instruction (PSSI) Option", draft-bonica-6man-seg-end-opt-04 (work in progress), July 2019.
- [I-D.bonica-6man-vpn-dest-opt]
Bonica, R., Kamite, Y., Lenart, C., So, N., Xu, F., Presbury, G., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The Per-Path Service Instruction (PPSI) Option", draft-bonica-6man-vpn-dest-opt-06 (work in progress), July 2019.
- [I-D.bonica-lsr-crh-isis-extensions]
Kaneriya, P., Shetty, R., Hegde, S., and R. Bonica, "IS-IS Extensions To Support The IPv6 Compressed Routing Header (CRH)", draft-bonica-lsr-crh-isis-extensions-00 (work in progress), May 2019.
- [I-D.ssangli-idr-bgp-vpn-srv6-plus]
Ramachandra, S. and R. Bonica, "BGP based Virtual Private Network (VPN) Services over SRv6+ enabled IPv6 networks", draft-ssangli-idr-bgp-vpn-srv6-plus-02 (work in progress), July 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

15.2. Informative References

- [I-D.filsfils-spring-net-pgm-extension-srv6-usid]
Filsfils, C., Camarillo, P., Cai, D., Jiang, Z., daniel.voyer@bell.ca, d., Shawky, A., Leymann, N., Steinberg, D., Zandi, S., Dawra, G., Meilik, I., Uttaro, J., Jalil, L., So, N., Fiumano, M., and M. Khaddam, "Network Programming extension: SRv6 uSID instruction", draft-filsfils-spring-net-pgm-extension-srv6-usid-02 (work in progress), August 2019.
- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-24 (work in progress), October 2019.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-04 (work in progress), October 2019.
- [I-D.li-spring-compressed-srv6-np]
Li, Z., Li, C., Peng, S., Wang, Z., and B. Liu, "Compressed SRv6 Network Programming", draft-li-spring-compressed-srv6-np-00 (work in progress), July 2019.

- [I-D.mirsky-6man-unified-id-sr]
Cheng, W., Mirsky, G., Peng, S., Aihua, L., Wan, X., and C. Wei, "Unified Identifier in IPv6 Segment Routing Networks", draft-mirsky-6man-unified-id-sr-03 (work in progress), July 2019.
- [RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, DOI 10.17487/RFC2151, June 1997, <<https://www.rfc-editor.org/info/rfc2151>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

Authors' Addresses

Ron Bonica
Juniper Networks
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Shraddha Hegde
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: shraddha@juniper.net

Yuji Kamite
NTT Communications Corporation
3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Email: y.kamite@ntt.com

Andrew Alston
Liquid Telecom
Nairobi
Kenya

Email: Andrew.Alston@liquidtelecom.com

Daniam Henriques
Liquid Telecom
Johannesburg
South Africa

Email: daniam.henriques@liquidtelecom.com

Luay Jalil
Verizon
Richardson, Texas
USA

Email: luay.jalil@one.verizon.com

Joel Halpern
Ericsson
P. O. Box 6049
Leesburg, Virginia 20178
USA

Email: joel.halpern@ericsson.com

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

Gang Chen
Baidu
No.10 Xibeiwang East Road Haidian District
Beijing 100193
P.R. China

Email: phdgang@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2020

J. Dong
Huawei Technologies
S. Bryant
Futurewei Technologies
T. Miyasaka
KDDI Corporation
Y. Zhu
China Telecom
F. Qin
Z. Li
China Mobile
October 16, 2019

Segment Routing for Enhanced VPN Service
draft-dong-spring-sr-for-enhanced-vpn-05

Abstract

Enhanced VPN (VPN+) is an enhancement to VPN services to enable it to support the needs of new applications, particularly applications that are associated with 5G services. These applications require better isolation from both control and data plane's perspective and have more stringent performance requirements than can be provided with overlay VPNs. The characteristics of an enhanced VPN as perceived by its tenant needs to be comparable to those of a dedicated private network. This requires tight integration between the overlay VPN and the underlay network topology and resources in a scalable manner. An enhanced VPN may form the underpinning of 5G network slicing, but will also be of use in its own right. This document describes how to use segment routing based mechanisms to provide the enhanced VPN service with the required network topology and resources. The overall mechanism of providing segment routing based enhanced VPN service is also described. The proposed mechanism is applicable to both segment routing with MPLS data plane (SR-MPLS) and segment routing with IPv6 data plane (SRv6).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Segment Routing with Topology and Resource Awareness | 4 |
| 2.1. SR-MPLS | 5 |
| 2.2. SRv6 | 6 |
| 3. Control Plane Considerations | 7 |
| 4. Procedures | 7 |
| 4.1. Virtual Network Topology and Resource Computation | 8 |
| 4.2. Network Resource and SID Allocation | 9 |
| 4.3. Construction of SR Virtual Networks | 11 |
| 4.4. VPN Service to SR Virtual Network Mapping | 12 |
| 4.5. Virtual Network Visibility to Customer | 13 |
| 5. Benefits of the Proposed Mechanism | 13 |
| 6. Service Assurance | 14 |
| 7. IANA Considerations | 14 |
| 8. Security Considerations | 15 |
| 9. Contributors | 15 |
| 10. Acknowledgements | 15 |
| 11. References | 15 |
| 11.1. Normative References | 15 |

| | |
|--|----|
| 11.2. Informative References | 16 |
| Authors' Addresses | 19 |

1. Introduction

Driven largely by needs arising from the 5G mobile network design, the concept of network slicing has gained traction [NGMN-NS-Concept] [TS23501][TS28530] [BBF-SD406]. In [TS23501], Network Slice is defined as "a logical network that provides specific network capabilities and network characteristics", and Network Slice Instance is defined as "A set of Network Function instances and the required resources (e.g. compute, storage and networking resources) which form a deployed Network Slice". Network slicing requires to partition the physical network to several pieces to provide each network slice with the required networking, computing, and storage resources and functions to meet the requirement of slice tenants.

As specified in [I-D.ietf-teas-enhanced-vpn], a transport network slice is a virtual (logical) network with a particular network topology and a set of shared or dedicated network resources, which are used to provide the network slice consumer with the required connectivity, appropriate isolation and specific Service Level Agreement (SLA). In order to meet the requirement of transport network slicing, there is a need to create virtual networks with enhanced characteristics. Such a virtual network can provide a customized network topology, some degree of isolation and performance guarantee to meet the slice tenant's requirement. Additionally, a network slice tenant may ask for some level of control to their virtual network, e.g. to customize the service paths in the network slice.

The enhanced VPN service (VPN+) as described in [I-D.ietf-teas-enhanced-vpn] is targeted at new applications which require better isolation from both control plane and data plane's perspective, and have more stringent performance requirements than can be provided with existing overlay VPNs. An enhanced VPN may form the underpinning of network slicing, but will also be of use in its own right.

Although a VPN can be associated with a set of dedicated RSVP-TE [RFC3209] LSPs with bandwidth reservation to provide some level of guarantee to service performance, such mechanisms would introduce per-VPN per-path states in the network, which is known to have scalability issues [RFC5439] and has not been widely adopted in production networks.

Segment Routing (SR) [RFC8402] specifies a mechanism to steer packets through an ordered list of segments. A segment is often referred to

by its Segment Identifier (SID). With SR, explicit source routing can be achieved without introducing per-path state into the network. Compared with RSVP-TE, currently SR does not have the capability of reserving or identifying a particular set of network resources reserved for particular services or customers. Although a centralized controller can have a global view of network state and can provision different services onto different SR paths, in packet forwarding it still relies on traditional DiffServ QoS mechanism [RFC2474] [RFC2475] to provide coarse-grained traffic differentiation in the network. While such kind of mechanism may be sufficient for some types of services, it cannot meet the stringent requirement of some enhanced VPN services which need to be isolated from other services in the network. Also the number of such enhance VPN services would be larger than the number of classes in DiffServ QoS.

This document extends the SR paradigm by introducing additional Segment Identifiers (SIDs) to represent different virtual network topologies and the corresponding set of network resources allocated to the virtual networks on each network segment. A group of SR SIDs can be used to specify the customized topology of an enhanced VPN, and can be further used to steer the service traffic to be processed with the corresponding set of network resources. The proposed mechanism is applicable to SR with both MPLS data plane (SR-MPLS) and IPv6 data plane (SRv6). The overall mechanism of providing segment routing based enhanced VPN is also described.

2. Segment Routing with Topology and Resource Awareness

In order to support enhanced VPN service, the overlay VPNs need to be integrated with the underlay network in terms of network topology and network resources. More specifically, enhanced VPNs need to be mapped to different virtual topologies to provide customized connectivity, and different set of network resources may be allocated to different enhanced VPNs, or different groups of enhanced VPNs, to meet the diverse performance requirement. When SR is used as the data plane to enable enhanced VPNs, it is necessary that the SR service paths are computed within a particular virtual network topology, and are instantiated with a particular set of network resources.

In the segment routing architecture [RFC8402], several types of segments are defined to represent either topological or service instructions. A topological segment can be a node segment or an adjacency segment. A service segment may be associated with specific service function for service chaining purpose. This document introduces additional SR segments which can be associated with particular network topology and particular set of network resources.

This section describes the mechanisms to identify the associated virtual network topology and resource information with the two SR data plane instantiations: SR-MPLS and SRv6.

2.1. SR-MPLS

In SR-MPLS [I-D.ietf-spring-segment-routing-mpls], IGP Adjacency Segment (Adj-SID) is an IGP-segment attached to a unidirectional adjacency or a set of unidirectional adjacencies. IGP Node segment is an IGP-Prefix segment that identifies a specific router (e.g., a loopback). In [I-D.ietf-idr-bgpls-segment-routing-epe], PeerAdj SID is used as instruction to steer over a specific local interface towards a specific peer node in a peering Autonomous System (AS). These types of SIDs can be extended to represent both topological elements and the resources allocated on a particular network element.

For one particular IGP link, multiple Adj-SIDs SHOULD be allocated, each of which is associated with a particular virtual network topology, and MAY represent a subset of link resources. Several approaches can be used to partition the link resource, such as logical sub-interfaces, dedicated queues, etc. The detailed mechanism of resource partitioning is out of scope of this document. Similarly, for one particular IGP node, multiple node-SIDs SHOULD be allocated, each of which is associated with a specific virtual network topology, and may represent a subset of the node resource (e.g. the processing resources). For one inter-domain link, multiple BGP PeerAdj SIDs [I-D.ietf-idr-bgpls-segment-routing-epe] can be allocated, each of which is associated with a specific virtual network topology which spans multiple domains, and MAY represent a subset of link resource on the inter-domain link. Note that this per-segment resource allocation complies to the SR paradigm, which avoids introducing per-path state into the network.

A group of adj-SIDs and node-SIDs associated with the same virtual network can be used to construct the SR SID-lists (either strict or loose) to steer the traffic of a particular enhanced VPN service. This group of SIDs MAY also represent the set of network resources which are reserved for a specific enhanced VPN, or a group of enhanced VPNs.

In data packet forwarding, the adj-SID and node-SID are used to identify the virtual network the packet belongs to, so that a virtual network specific next-hop can be determined. The adj-SIDs MAY be used to steer traffic of different enhanced VPNs into different set of link resources. The node SIDs MAY be used to steer traffic of different enhanced VPNs into different set of node resources. When a node-SID is used in the SID-list to build an SR loose path, the transit nodes use the node-SID to identify the virtual network, and

MAY process the packet using the local resources allocated for the corresponding virtual network. Note in this case, Penultimate Hop Popping (PHP) [RFC3031] MUST be disabled.

This mechanism requires to allocate additional node-SIDs and adj-SIDs for each virtual network (network slice). As the number of virtual networks increases, the number of SIDs would increase accordingly. It is expected that this mechanism is applicable to a network with a limited number of network slices.

2.2. SRv6

As specified in [I-D.ietf-spring-srv6-network-programming], an SRv6 Segment Identifier (SID) is a 128-bit value which consists of a locator (LOC) and a function (FUNCT), optionally it may also contain additional arguments (ARG) immediately after the FUNCT. The LOC of the SID is routable and leads to the node which instantiates that SID, which means the LOC can be parsed by all nodes in the network. The FUNCT part of the SID is an opaque identification of a local function bound to the SID, which means the FUNCT and ARG parts can only be parsed by the node which instantiates that SID.

In order to support multiple virtual networks in a SRv6 network, all the nodes (including the edge nodes and transit nodes) belonging to one particular virtual network MUST have a consistent view of the virtual network and performs consistent computation and forwarding behavior to comply to the network topology and resource constraints. A node which participates in multiple virtual networks MUST be able to distinguish packets which belong to different virtual networks.

Taking the above into consideration, for a particular network node, multiple SRv6 LOCs SHOULD be allocated, each of which is associated with a virtual network topology, and MAY represent a subset of the network resources associated with the virtual network. The SRv6 SIDs of a particular virtual network SHOULD be allocated from the SID space using the virtual network specific LOC as the prefix. These SRv6 SIDs can be used to represent virtual network specific local functions.

A group of SRv6 SIDs associated with the same virtual network can be used to construct the SR SID-lists (either strict or loose) to steer the traffic of a particular enhanced VPN service. This group of SIDs MAY also represent the set of network resources which are reserved for a particular enhanced VPN, or a group of enhanced VPNs.

In data packet forwarding, the LOC part of SRv6 SID is used by transit nodes to identify the virtual network the packet belongs to, so that a virtual network specific next-hop can be determined. The

LOC MAY also be used to indicate the set of local network resources on the transit nodes to be used for the forwarding of the received packet. The SRv6 segment endpoint nodes use the virtual network specific SRv6 SID to identify the virtual network the packet belongs to, and the particular local function to perform on the received packet. The local SRv6 SID MAY also be used to identify the set of network resource to be used for executing the local function.

This mechanism requires to allocate additional SRv6 Locators and SIDs for each virtual network (network slice). As the number of virtual networks increases, the number of Locators and SIDs would increase accordingly. It is expected that this mechanism is applicable to a network with a limited number of network slices.

3. Control Plane Considerations

The mechanism described in this document makes use of a centralized controller that collects the information about the network (configuration, state, routing databases, etc.) as well as the service information (traffic matrix, performance statistics, etc.). The controller is also responsible for the centralized computation and optimization of the SR paths within virtual networks for different enhanced VPNs. The SR SIDs can be either explicitly provisioned by the controller, or dynamically allocated by network nodes then reported to the controller. The interaction between the controller and the network nodes can be based on PCEP [RFC5440], Netconf/YANG [RFC6241] [RFC7950] and BGP-LS [RFC7752]. In some scenarios, extensions to some of these protocols is needed, which are out of the scope of this document and will be specified in separate documents.

A distributed control plane can be used for the collection and distribution of the network topology and state information of the virtual networks among network nodes. Distributed route computation for services of a particular enhanced VPN is also needed. The IGP extensions for SR based enhanced VPN are specified in [I-D.dong-lsr-sr-enhanced-vpn].

4. Procedures

This section describes the procedures of provisioning enhanced VPN service in SR based virtual networks.

According to the requirement of an enhanced VPN service, a centralized network controller calculates a subset of the physical network topology to support the enhanced VPN. Within this topology, the set of network resources needed on each network element is also determined. This network topology and the set of network resources

together constitute a virtual network (or network slice). Depending on the service requirement, the network topology and resource can be dedicated for a particular enhanced VPN, or they can be shared among multiple enhanced VPNs.

Following the segment routing paradigm, the network topology and resource are represented using a group of dedicated SIDs. The group of node-SIDs and adj-SIDs allocated for a virtual network will be used by network nodes and the network controller to build a SR based virtual network, which is used as the underlay of the enhanced VPN service. The extensions to IGP protocol to distribute the SIDs and the associated resources allocated for a virtual network are specified in [I-D.dong-lsr-sr-enhanced-vpn].

Suppose tenant A requests for an enhanced VPN service from the network operator. The requirement is that service of tenant A does not experience unexpected interference from other services in the same network, such as other tenants' VPN services, or the non-VPN services in the network. The detailed requirements can be described with characteristics such as the following:

- o Service topology: the service sites and the connectivity between them.
- o Service bandwidth: the bandwidth requirement between service sites.
- o Isolation: the level of isolation from other services in the network.
- o Reliability: whether fast local repair or end-to-end protection is needed or not.
- o Latency: the maximum latency for specific service between specific service sites.
- o Visibility: the customer may want to have some form of visibility of the virtual network delivering the service.

4.1. Virtual Network Topology and Resource Computation

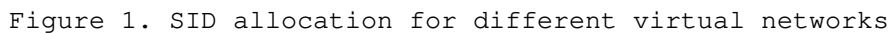
As described in section 4, a centralized network controller is responsible for the computation of a virtual network for the provisioning of enhanced VPNs. The controller collects the information of network connectivity, network resources, network performance and other relevant network states of the underlay network. This can be done using either IGP [RFC5305] [RFC3630] [RFC7471] [RFC7810] or BGP-LS [RFC7752] [RFC8571].

Based on the information collected from the underlay network, controller obtains the underlay network topology and the information about the allocated and available network resources. When an enhanced VPN service request is received from a tenant, the controller computes the subset of the network topology, along with set of the resources needed on each network element (e.g. links and nodes) in the topology to meet the tenant's service requirements, whilst maintaining the needs of the existing tenants that are using the same network. The subset of network topology and resource constitute a virtual network, which will be used as the underlay of the requested enhanced VPN.

4.2. Network Resource and SID Allocation

According to the result of virtual network computation, network controller instructs the network devices involved in the virtual network to allocate the required network resources for the enhanced VPN. This can be done with either PCEP [RFC5440] or Netconf/YANG [RFC6241] [RFC7950] with necessary extensions. The network resources are allocated in a per-segment manner. In addition, a set of dedicated SIDs, e.g. node-SIDs and adj-SIDs are allocated to represent the virtual network and the network resources allocated on each network segment for this virtual network.

In the underlay forwarding plane, there can be multiple ways of partitioning and allocating a set of network resource to a virtual network. For example, [FLEXE] may be used to partition the link resource into different sub-channels to achieve hard isolation between each other. The candidate data plane technologies to support enhanced VPN can be found in [I-D.ietf-teas-enhanced-vpn]. The SR SIDs are used as a network layer abstraction for various network resource partitioning and allocation mechanisms in the underlay forwarding plane.



[Page 10]

each virtual network, in this case it can be used by the transit nodes to steer different service traffic into different set of local network resources in the forwarding plane.

In Figure 1, the notation `x:nnnn:y` means that in virtual network `x`, the adj-SID `nnnn` will steer the packet over a link which has bandwidth `y` reserved for that virtual network. Thus the note `r:1002:1G` in link C->D says that the red virtual network has a reserved bandwidth of 1Gb/s on link C->D, and will be used by packets arriving at node C with an adj-SID 1002 at the top of the label stack.

4.3. Construction of SR Virtual Networks

To make both the network controller and network nodes aware of the information of the virtual networks created in the network, each network node MUST advertise the virtual networks it participates, together with the set of SIDs and the associated resource attributes both into the network and to the controller. This can be achieved by means such as IGP extensions [I-D.dong-lsr-sr-enhanced-vpn], BGP-LS [RFC7752] with necessary extensions, NETCONF/YANG [RFC6241] [RFC7950].

Based on the collected information of network topology, network resource and SIDs information, both the controller and network nodes are able to construct the SR virtual network and generate the forwarding entries of each virtual network based on the node-SIDs and adj-SIDs allocated for each virtual network. Unlike classic segment routing in which network resources are always shared by all the services and tenants, different SR virtual networks can be associated with different set of resource allocated in the underlay network, so that they can be used to meet the service requirement of enhanced VPNs and provide the required isolation from other services in the same network.

Figure 2 shows the SR based virtual networks created in the network in Figure 1.

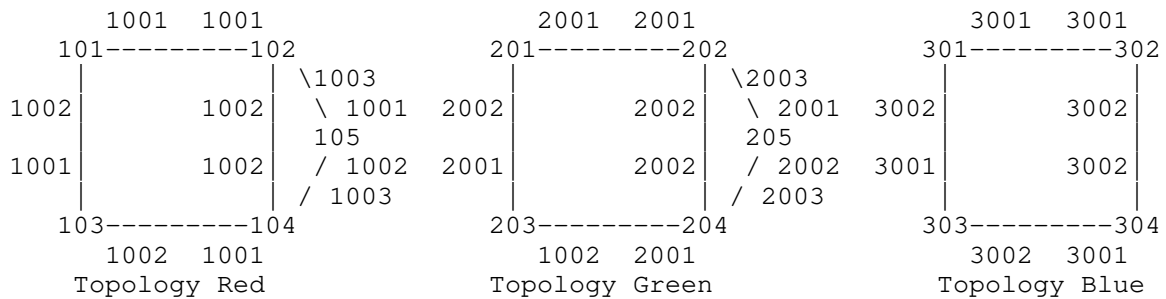


Figure 2. SR based virtual networks with different groups of SIDs

In each SR virtual network, SR path is computed within the virtual network, taking its network topology and resource as constraints. The SR path can be an explicit path instantiated using SR policy [I-D.ietf-spring-segment-routing-policy], in which the segment-list is built with the SIDs allocated to the virtual network. The service path can also be an IGP computed path associated with a particular node-SIDs of the virtual network. Different service paths in the same virtual network would share the network resources allocated to the virtual network.

For example, to create an explicit path A-B-D-E in the virtual network red in Figure 2, the SR segment list encapsulated in the service packet would be (1001, 1002, 1003). For the same explicit path A-B-D-E in virtual network green, the SR segment list would be (2001, 2002, 2003). In the case where we wish to construct a loose path A-D-E in virtual network green, the service packet SHOULD be encapsulated with the SR segment list (201, 204, 205). At node A, the packet can be sent towards D via either node B or C using the link and node resources allocated for virtual network green. At node D the packet is forwarded to E using the link and node resource allocated for virtual network green. Similarly, a packet to sent via loose path A-D-E in virtual network red would be encapsulated with segment list (101, 104, 105). In the case where an IGP computed path can meet the service requirement, the packet can be simply encapsulated with the node SID of egress node E in the corresponding virtual network.

4.4. VPN Service to SR Virtual Network Mapping

The enhanced VPN services can be provisioned using the customized SR virtual networks as the underlay network. Different enhanced VPNs may be provisioned in different SR virtual networks, each of which would use the network resources allocated to a particular virtual network, so that they will not interfere with each other. In another case, a group of enhanced VPNs which have similar characteristic and

requirement can be provisioned in the same virtual network, in this case the network resources allocated to the virtual network are shared by this set of enhanced VPNs, but will not be shared with other services in the network.

4.5. Virtual Network Visibility to Customer

The tenants of enhanced VPNs may request different granularity of visibility to the network which deliver the service. Depending on the requirement, the network can be exposed to the tenant either as a virtual network, or a set of computed paths with transit nodes, or simply the abstract connectivity between endpoints without any path information. The visibility can be delivered through different possible mechanisms, such as IGPs (e.g. IS-IS, OSPF) or BGP-LS. In addition, network operator may want to restrict the visibility of the information it delivers to the tenant by either hiding the transit nodes between sites (and only delivering the endpoints connectivity), or by hiding portions of the transit nodes (summarizing the path into fewer nodes). Mechanisms such as BGP-LS allow the flexibility of the advertisement of aggregated virtual network information.

5. Benefits of the Proposed Mechanism

The proposed mechanism provides several key characteristics:

- o **Flexibility:** Multiple customized virtual networks can be created in a shared network to meet different tenants' connectivity and service requirement. Each tenant is only aware of the topology and attributes of his own virtual network, and provision services on the virtual network instead of the physical network. This provides an efficient mechanism to support network slicing.
- o **Isolation:** Each virtual network can have independent SR path computation and instantiation. In addition, a virtual network can be associated with a set of network resources, which can avoid resource competition and performance interference from other services in the network. The proposed mechanism also allows resource sharing between different services in the same enhanced VPN, or between a group of enhanced VPNs which are provisioned in the same virtual network. This gives the operator and the enhanced VPN tenants the flexibility in network planning and service provisioning. The performance of critical services in a particular enhanced VPN can be further ensured using the mechanisms defined in [DetNet].
- o **Scalability:** The proposed mechanism seeks to achieve a balance between the state limitations of traditional end-to-end TE mechanism and the lack of resource awareness in basic segment

routing. Following the segment routing paradigm, network resources are allocated to network segments in the virtual networks, thus there is no per-flow state introduced in the network. Operator can choose the granularity of resource allocation to network segments. In network segments where resource is scarce such that the service requirement may not always be met, the SR approach can be used to allocate specific resources to the segment in a particular virtual network. By contrast, in other segment of the network where resource is considered plentiful, the resource may be shared between a number of virtual networks. The decision to do this is in the hands of the operator. Because of the segmented nature of the virtual network, resource aggregation is easier and more flexible than RSVP-TE based approach.

6. Service Assurance

In order to provide service assurance for enhanced VPNs, it is necessary to instrument the network at multiple levels. The network operator needs to ascertain that the underlay is operating correctly. A tenant needs to ascertain that their services are operating correctly. In principle these can use existing techniques. These are well known problems and solutions either exist or are in development to address them.

New work is needed to instrument the virtual networks that are created for the enhanced VPNs. Such instrumentation needs to operate without causing disruption to other services using the network. Given the sensitivity of some applications, care needs to be taken to ensure that the instrumentation itself does not cause disruption either to the service being instrumented or to other services. In case of failure or performance degradation of a service path in a particular virtual network, it is necessary that either local protection or end-to-end protection mechanism is used to switch to another path which could meet the service performance requirement and does not impact other services in the network.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

The normal security considerations of VPNs are applicable and it is assumed that industry best practise is applied to an enhanced VPN.

The security considerations of segment routing are applicable and it is assumed that these are applied to an enhanced VPN that uses SR based virtual networks.

Some applications of enhanced VPNs are sensitive to packet latency; the enhanced VPNs provisioned to carry their traffic have latency SLAs. By disrupting the latency of such traffic an attack can be directly targeted at the customer application, or can be targeted at the network operator by causing them to violate their SLA, triggering commercial consequences. Dynamic attacks of this sort are not something that networks have traditionally guarded against, and networking techniques need to be developed to defend against this type of attack. By rigorously policing ingress traffic and carefully provisioning the resources provided to critical services this type of attack can be prevented. However care needs to be taken when providing shared resources, and when the network needs to be reconfigured as part of ongoing maintenance or in response to a failure.

The details of the underlay MUST NOT be exposed to third parties, to prevent attacks aimed at exploiting a shared resource.

9. Contributors

The following people contributed to the content of this document.

10. Acknowledgements

The authors would like to thank Mach Chen, Zhenbin Li, Stefano Previdi, Charlie Perkins, Bruno Decraene and Loa Andersson for the valuable discussion and suggestions to this document.

11. References

11.1. Normative References

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

11.2. Informative References

- [BBF-SD406] "BBF SD-406: End-to-End Network Slicing", 2016, <<https://wiki.broadband-forum.org/display/BBF/SD-406+End-to-End+Network+Slicing>>.
- [DetNet] "DetNet WG", 2016, <<https://datatracker.ietf.org/wg/detnet>>.
- [FLEXE] "Flex Ethernet Implementation Agreement", March 2016, <<http://www.oiforum.com/wp-content/uploads/OIF-FLEXE-01.0.pdf>>.
- [I-D.dong-lsr-sr-enhanced-vpn] Dong, J. and S. Bryant, "IGP Extensions for Segment Routing based Enhanced VPN", draft-dong-lsr-sr-enhanced-vpn-01 (work in progress), October 2018.
- [I-D.ietf-6man-segment-routing-header] Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-25 (work in progress), October 2019.
- [I-D.ietf-idr-bgpls-segment-routing-epe] Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-19 (work in progress), May 2019.

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-04 (work in progress), October 2019.
- [I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Networks (VPN+) Service", draft-ietf-teas-enhanced-vpn-03 (work in progress), September 2019.
- [NGMN-NS-Concept]
"NGMN NS Concept", 2016, <https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <<https://www.rfc-editor.org/info/rfc2475>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.

- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5439] Yasukawa, S., Farrel, A., and O. Komolafe, "An Analysis of Scaling Issues in MPLS-TE Core Networks", RFC 5439, DOI 10.17487/RFC5439, February 2009, <<https://www.rfc-editor.org/info/rfc5439>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <<https://www.rfc-editor.org/info/rfc7810>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.
- [TS23501] "3GPP TS23.501", 2016, <<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144>>.
- [TS28530] "3GPP TS28.530", 2016, <<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3273>>.

Authors' Addresses

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Stewart Bryant
Futurewei Technologies

Email: stewart.bryant@gmail.com

Takuya Miyasaka
KDDI Corporation

Email: ta-miyasaka@kddi.com

Yongqing Zhu
China Telecom

Email: zhuyq.gd@chinatelecom.cn

Fengwei Qin
China Mobile

Email: qinfengwei@chinamobile.com

Zhenqiang Li
China Mobile

Email: li_zhenqiang@hotmail.com

SPRING Working Group
Internet-Draft
Intended status: Informational
Expires: July 22, 2021

J. Dong
Huawei Technologies
S. Bryant
Futurewei Technologies
T. Miyasaka
KDDI Corporation
Y. Zhu
China Telecom
F. Qin
Z. Li
China Mobile
F. Clad
Cisco Systems
January 18, 2021

Segment Routing based Virtual Transport Network (VTN) for Enhanced VPN
draft-dong-spring-sr-for-enhanced-vpn-13

Abstract

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an ordered list of instructions, called "segments". A segment can represent topological or service based instructions. A segment can further be associated with a set of network resources used for executing the instruction. Such a segment is called resource-aware segment.

Resource-aware Segment Identifiers (SIDs) may be used to build SR paths with a set of reserved network resources. In addition, a group of resource-aware SIDs may be used to build SR based virtual underlay networks, which has customized network topology and resource attributes required by one or a group of customers and/or services. Such virtual networks are the SR instantiations of Virtual Transport Networks (VTNs).

This document describes a suggested use of resource-aware SIDs to build SR based VTNs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 22, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Resource-Aware SIDs for VTN | 3 |
| 2.1. SR-MPLS based VTN | 4 |
| 2.2. SRv6 based VTN | 4 |
| 2.3. VTN Identification | 5 |
| 2.4. Scalability Considerations | 5 |
| 3. Procedures | 5 |
| 3.1. VTN Topology and Resource Planning | 6 |
| 3.2. VTN Network Resource and SID Allocation | 7 |
| 3.3. Construction of SR based VTNs | 9 |
| 3.4. Mapping Service to SR based VTN | 11 |
| 3.5. VTN Visibility to Customer | 11 |
| 4. Characteristics of SR based VTN | 12 |
| 5. Service Assurance of VTN | 13 |
| 6. IANA Considerations | 13 |
| 7. Security Considerations | 13 |
| 8. Contributors | 14 |
| 9. Acknowledgements | 14 |
| 10. References | 14 |
| 10.1. Normative References | 14 |
| 10.2. Informative References | 14 |
| Authors' Addresses | 17 |

1. Introduction

Segment Routing (SR) [RFC8402] specifies a mechanism to steer packets through an ordered list of segments. A segment is referred to by its Segment Identifier (SID). With SR, explicit source routing can be achieved without introducing per-path state into the network.

[I-D.ietf-spring-resource-aware-segments] proposes to extend SR by associating SIDs with network resource attributes (e.g. bandwidth, processing or storage resources). These resource-aware SIDs retain their original functionality, with the additional semantics of identifying the set of network resources available for the packet processing action. On a network segment, multiple resource-aware SIDs may be allocated, each of which is associated with a subset of network resources assigned to meet the requirements of one or a group of customers and/or services.

Once allocated, Resource-aware SIDs can be used to build SR paths with a set of reserved network resources. In addition, a group of resource-aware SIDs may be used to build SR based virtual networks, which has customized network topology and resource attributes required by one or a group of customers and/or services. Such virtual networks are the SR instantiations of Virtual Transport Networks (VTNs) as defined in [I-D.ietf-teas-enhanced-vpn], and can be used to enable the enhanced VPN (VPN+) services.

This document describes a suggested use of resource-aware SIDs to build SR based VTNs. Although the procedure is illustrated using SR-MPLS, the proposed mechanism is applicable to both SR over MPLS data plane (SR-MPLS) and SR over IPv6 data plane (SRv6).

2. Resource-Aware SIDs for VTN

A VTN is a virtual underlay network which has a specific network topology and a subset of network resources allocated from the physical network.

When SR is used as the data plane to construct VTNs in the network, it is necessary to compute and instantiate the SR paths with the topology and/or algorithm constraints of the VTN, and steer the traffic to only use the set of network resources allocated to the VTN.

Based on the resource-aware segments defined in [I-D.ietf-spring-resource-aware-segments], a group of resource-aware SIDs can be allocated to represent the network segments of one VTN. These resource-aware SIDs are associated with the group of network resources allocated to the VTN on network nodes and links which participate in the VTN. These resource-aware SIDs can also identify

the network topological or functional instructions associated with the VTN.

The resource-aware SIDs may be allocated either by a centralized network controller or by network nodes. The control plane mechanisms for advertising the resource-aware SIDs for VTNs can be based on [RFC4915], [RFC5120] and [I-D.ietf-lsr-flex-algo] with necessary extensions. This is further described in section 3.3.

2.1. SR-MPLS based VTN

This section describes a mechanism of allocating resource-aware SIDs to SR-MPLS based VTNs.

For one IGP link, multiple Adj-SIDs are allocated, each of which is associated with a VTN that link participates in, and represents a subset of the link resources allocated to the VTN. For one IGP node, multiple prefix-SIDs are allocated, each of which is associated with a VTN which the node participates in, and identifies the set of network resources allocated to the VTN on network nodes which participate in the VTN. These set of resources will be used to process packets which have the resource-aware SIDs as the active segment.

In the case of multi-domain VTNs, on an inter-domain link, multiple BGP peering SIDs [I-D.ietf-idr-bgppls-segment-routing-epe] are allocated, each of which is associated with a VTN which spans multiple domains, and represents a subset of resources allocated on the inter-domain link.

2.2. SRv6 based VTN

This section describes a mechanism of allocating resource-aware SRv6 Locators and SIDs to SRv6 based VTNs.

For a network node, multiple SRv6 Locators are allocated, each of which is associated with a VTN the node participates in, and identifies the set of network resources allocated to the VTN on network nodes which participate in the VTN. The SRv6 SIDs associated with a VTN are allocated from the SID space using the VTN-specific Locator as the prefix. These SRv6 SIDs can be used to represent VTN-specific SRv6 functions, and can identify the set of resources used by network nodes to process packets.

2.3. VTN Identification

In a simple case, each VTN can be mapped to a unique topology or algorithm. Then the VTNs can be distinguished by the topology ID or algorithm ID in control plane, and the resource-aware SIDs associated with a VTN can be identified using the <topology, algorithm> tuple as described in [RFC8402]. The number of VTNs supported in a network relies on the number of topologies or algorithms supported.

In a more complicated case, multiple VTNs may be mapped to the same <topology, algorithm> tuple, while each is allocated with a separate set of network resources. Then a new VTN Identifier (VTN-ID) in the control plane is needed to identify the VTN. The resource-aware SIDs associated with different VTNs can be distinguished using VTN-IDs.

In the data plane, The resource-aware SIDs are used to identify the VTN, and are also used to determine the forwarding instructions and the set of network resources used for the packet processing action.

2.4. Scalability Considerations

Since multiple VTNs can be created in a network, and each VTN is allocated with a group of resource-aware SIDs, the mechanism of SR based VTNs increases the number of SIDs and SRv6 Locators needed in a network. There may be some concern, especially about the SR-MPLS prefix-SIDs, which are allocated from the Segment Routing Global Block (SRGB). The amount of network state will also increase accordingly. However, based on the SR paradigm, resource-aware SIDs and the associated network state are allocated and maintained per VTN, thus per-path network state is avoided in the SR network.

3. Procedures

This section describes possible procedures for creating SR based VTNs and the corresponding forwarding tables and entries. Although it is illustrated using SR-MPLS, the proposed mechanism is applicable to both SR-MPLS and SRv6.

Suppose a virtual network is requested by some customer or service. One of the basic requirement is that customer or service is allocated with some dedicated network resource, so that it does not experience unexpected interference from other services in the same network. Other possible requirements may include the required topology, bandwidth, latency, reliability, etc.

According to the received requirement, a centralized network controller calculates a subset of the underlay network topology to support the service. With this topology, the set of network

resources required on each network element is also determined. The subset of network topology and network resources are the two major characteristics of a VTN. Depending on the service requirement, the network topology and network resource of this VTN can be dedicated for an individual customer or service, or can be shared by a group of customers and/or services.

Based on the mechanisms described in section 2, a group of resource-aware SIDs can be allocated for the VTN. With SR-MPLS, it is a group of prefix-SIDs and adj-SIDs which are allocated to identify the network nodes and links in the VTN, and also identify the set of network resources allocated on these network nodes and links for the VTN. As the resource-aware SIDs can be allocated either by a centralized network controller or by the network nodes, control plane protocols such as IGP (e.g. IS-IS or OSPF) and BGP-LS can be used to distribute the SIDs and the associated resource and topology information of a VTN to other nodes in the same VTN and also to the controller, so that both the network nodes and the controller can generate the VTN-specific forwarding entries based on the resource-aware SIDs of the VTN. The detailed control plane mechanisms and possible extensions are described in separate documents and are out of the scope of this document.

3.1. VTN Topology and Resource Planning

A centralized network controller can be responsible for the planning of a VTN to meet the received service request. The controller needs to collect the information on network connectivity, network resources, network performance and any other relevant network states from the underlay network. This can be done using either IGP TE extensions such as [RFC5305] [RFC3630] [RFC7471] [RFC8570], and/or BGP-LS [RFC7752] [RFC8571], or any other form of control plane signaling.

Based on the information collected from the underlay network, the controller obtains the underlay network topology and the information about the allocated and available network resources. When a service request is received, the controller determines the subset of the network topology, and the set of the resources needed on each network segment (e.g. links and nodes) in the sub-topology to meet the service requirements, whilst maintaining the needs of the existing services that are using the same network. The subset of the network topology and network resources will be used to constitute a VTN, and will be used as the virtual underlay network of the requested service.

3.2. VTN Network Resource and SID Allocation

According to the result of VTN planning, the network controller instructs the set of network nodes involved to join the VTN and allocate the required set of network resources for the VTN. This may be done with PCEP [RFC5440], Netconf/YANG [RFC6241] [RFC7950] or with any other control or management plane mechanism with necessary extensions. Thus, the controller not only allocates the resources to the newly computed VTN but also keeps track of the remaining available resources in order to cope with subsequent VTN requests.

On each network node involved in the VTN, a set of network resources (e.g. link bandwidth) are allocated to the VTN. Such set of network resources can be dedicated for the processing of traffic in that VTN, and cannot be used by traffic in other VTNs. Note it is also possible that a group of VTNs may share a set of network resources on some network segments. A group of resource-aware SIDs, such as prefix-SIDs and adj-SIDs are allocated to identify both the network segments and the set of resources allocated on the network segments for the VTN. Such group of resource-aware SIDs, e.g. prefix-SIDs and adj-SIDs are used as the data plane identifiers of the nodes and links in the VTN.

In the underlying forwarding plane, there can be multiple ways of allocating a subset of network resources to a VTN. The candidate data plane technologies to support resource partitioning or reservation can be found in [I-D.ietf-teas-enhanced-vpn]. The resource-aware SIDs are considered as abstract data plane identifiers in the network layer, which can work with various network resource partitioning or reservation mechanisms in the underlying forwarding plane.

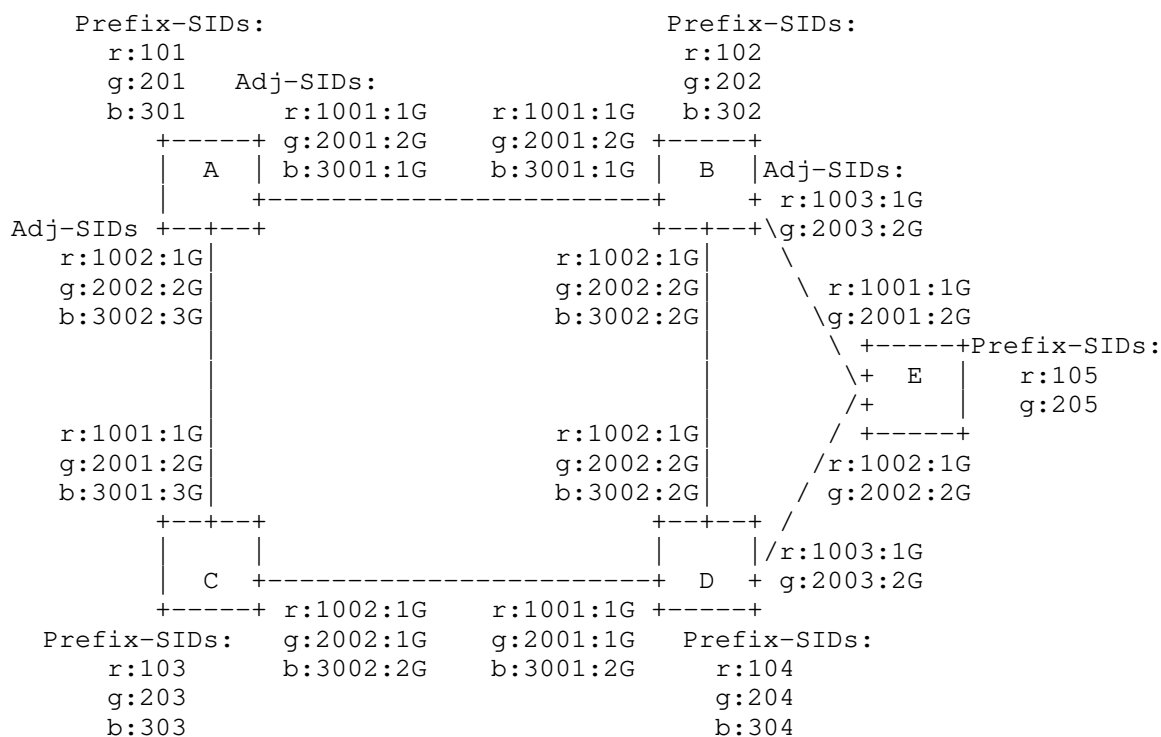


Figure 1. SID and resource allocation for multiple VTNs

Figure 1 shows an example of providing multiple VTNs in an SR based network. Note that the format of the SIDs in this figure is for illustration, both SR-MPLS and SRv6 can be used as the data plane. In this example, three VTNs: red (r) , green (g) and blue (b) are created to carry traffic of different customers or services. Both the red and green VTNs consist of nodes A, B, C, D, and E with all their interconnecting links, whilst the blue VTN only consists of nodes A, B, C and D with all their interconnecting links. Note that different VTNs may have a set of shared nodes and links. On each node, a resource-aware prefix-SID is allocated for each VTN it participates in. And on each link, a resource-aware adj-SID is allocated for each VTN it participates in.

In Figure 1, the notation x:nnnn:y means that in VTN x, the adj-SID nnnn will steer the packet over a link which has bandwidth y reserved for that VTN. For example, r:1002:1G in link C->D says that the VTN red has a reserved bandwidth of 1Gb/s on link C->D, and will be used by packets arriving at node C with an adj-SID 1002 at the top of the label stack. Similarly, on each node, a resource-aware prefix-SID is allocated for each VTN it participates in. Each resource-aware adj-

SID can be associated with a set of link resources (e.g. bandwidth) allocated to different VTNs, so that different adj-SIDs can be used to steer service traffic into different set of link resources in packet forwarding. A resource-aware prefix-SIDs in a VTN can be associated with the set of network resources allocated to this VTN on each involved network node and link. Thus the prefix-SIDs can be used to build loose SR path within a VTN, and can be used by the transit nodes to steer traffic into the set of local network resources allocated to the VTN.

3.3. Construction of SR based VTNs

The network controller needs to obtain the information of all the VTNs in the network it oversees, including the resource-aware SIDs and their associated network resources and topology information. Based on this information, the controller can have a global view of the VTN topology, network resources and the associated SIDs, so as to perform VTN-specific explicit path computation, taking both the topology and resource constraints of the VTN into consideration, and use the resource-aware SIDs to build the SID list for the explicit path. The controller may also compute the shortest paths in the VTN based on the resource-aware prefix-SIDs.

The network nodes also need to obtain the information of the VTNs they participate in, including the resource-aware SIDs and their associated network resources and topology information. Based on the collected information, the network nodes which are the headend of a path can perform VTN-specific path computation, and build the SID list using the collected resource-aware adj-SIDs and prefix-SIDs. The network nodes also need to generate the forwarding entries for the resource-aware prefix-SIDs in each VTN they participates in, and associate these forwarding entries with the set of local network resources (e.g. a set of bandwidth on the outgoing interface) allocated to the corresponding VTN.

Thus each network node needs to advertise the VTNs it participates in, the group of resource-aware SIDs allocated to each VTN, and the resource attributes (e.g. bandwidth) associated with the resource-aware SIDs in the network. Each resource-aware adj-SID is advertised with the set of associated link resources, and each resource-aware prefix-SID is advertised with the identifier of the associated VTN, as all the prefix-SIDs in a VTN are associated with the same set of network resources allocated to the VTN. Note as described in section 2.3, the VTN can be identified in the control plane either by existing IDs, or a new VTN ID.

The IGP mechanisms which reuse the existing IDs such as Multi-Topology [RFC5120] or Flex-Algo [I-D.ietf-lsr-flex-algo] as the

identifier of VTNs, and distribute the resource-aware SIDs and the associated topology and resource information are described in [I-D.xie-lsr-isis-sr-vtn-mt] and [I-D.zhu-lsr-isis-sr-vtn-flexalgo] respectively. The corresponding BGP-LS mechanisms which can be used to distribute both the intra-domain VTN information and the inter-domain VTN-specific link information to the controller are described in [I-D.xie-idr-bgppls-sr-vtn-mt] and [I-D.zhu-idr-bgppls-sr-vtn-flexalgo] respectively. Note that with these mechanisms, the number of VTNs supported relies on the number of topologies or algorithms supported.

The IGP mechanisms described in [I-D.dong-lsr-sr-enhanced-vpn] introduce a new VTN-ID in control plane, so that multiple VTNs can be mapped to the same <topology, algorithm> tuple, while each VTN can have different resource attributes. This allows flexible combination of network topology and network resources attributes to build a large number of VTNs with a relatively small number of topologies or algorithms. The corresponding BGP-LS mechanisms which can be used to distribute the intra-domain VTN information and the inter-domain VTN-specific link information to the controller are described in [I-D.dong-idr-bgppls-sr-enhanced-vpn].

Figure 2 shows the three SR based VTNs created in the network in Figure 1.

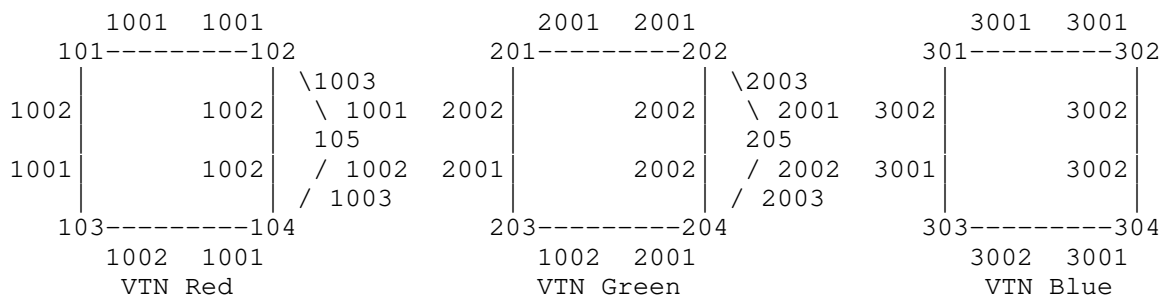


Figure 2. SR based VTNs with different groups of SIDs

For each SR based VTN, SR paths are computed within the VTN, taking the VTN topology and resources as constraints. The SR path can be an explicit path instantiated using SR policy [I-D.ietf-spring-segment-routing-policy], in which the SID-list is built only with the SIDs allocated to the VTN. The SR path can also be an IGP computed path associated with a prefix-SID or SRv6 End SID allocated by a node for the VTN, the IGP path computation is also based on the topology and/or algorithm constraints of the VTN. Different SR paths in the same VTN may use shared network resources when they use the same resource-aware SIDs allocated to the VTN,

while SR paths in different VTNs are steered to use different set of network resources even when they traverse the same network links or nodes. These VTN-specific SR paths need to be installed in the corresponding forwarding tables.

For example, to create an explicit path A-B-D-E in VTN red in Figure 2, the SR SID-list encapsulated in the service packet would be (1001, 1002, 1003). For the same explicit path A-B-D-E in VTN green, the SR segment list would be (2001, 2002, 2003). In the case where we wish to construct a loose path A-D-E in VTN green, the service packet SHOULD be encapsulated with the SR SID-list (201, 204, 205). At node A, the packet can be sent towards D via either node B or C using the network resources allocated by these nodes for VTN green. At node D the packet is forwarded to E using the link and node resource allocated for VTN green. Similarly, a packet to be sent via loose path A-D-E in VTN red would be encapsulated with segment list (101, 104, 105). In the case where an IGP computed path can meet the service requirement, the packet can be simply encapsulated with the prefix-SID of egress node E in the corresponding VTN.

3.4. Mapping Service to SR based VTN

Network services can be provisioned using SR based VTNs as the virtual underlay networks. For example, different services may be provisioned in different SR based VTNs, each of which would use the network resources allocated to the VTN, so that their data traffic will not interfere with each other. In another case, a group of services which have similar characteristics and requirements may be provisioned in the same VTN, in this case the network resources allocated to the VTN are only shared among this group of services, but will not be shared with other services in the network. The steering of service traffic to SR based VTNs can be based on either local policy or the mechanisms as defined in [I-D.ietf-spring-segment-routing-policy].

3.5. VTN Visibility to Customer

VTNs can be used by network operators to organize and split their network infrastructure into different virtual underlay networks for different customers or services. Some customers may also request different granularity of visibility to the VTN which is used to deliver the service. Depending on the requirement, VTN can be exposed to the customer either as a virtual network with both the edge nodes and the intermediate nodes, or a set of paths with some of the transit nodes, or simply a set of virtual connections between the endpoints without any transit node information. The visibility may be delivered through different possible mechanisms, such as IGPs (e.g. IS-IS, OSPF), BGP-LS or Netconf/YANG. On the other hand,

network operators may want to restrict the visibility of the underlay network information it delivers to the customer by either hiding the transit nodes between sites (and only delivering the endpoints connectivity), or by hiding portions of the transit nodes (summarizing the path into fewer nodes). Mechanisms such as BGP-LS allow the flexibility of the advertisement of aggregated virtual network information.

4. Characteristics of SR based VTN

The proposed mechanism provides several key characteristics:

- o Customization: Different customized VTNs can be created in a shared network to meet different customers' connectivity and service requirement. Each customer is only aware of the topology and attributes of his own VTN, and provision services on the VTN instead of the shared physical network. This provides an practical mechanism to support network slicing.
- o Resource Isolation: The computation and instantiation of SR paths in one VTN can be independent from other VTNs or other services in the network. In addition, a VTN can be associated with a set of dedicated network resources, which can avoid resource competition and performance interference from other VTNs or other services in the network. The proposed mechanism also allows resource sharing between different service flows of the same customer, or between a group of services which are provisioned in the same VTN. This gives the operators and the customers the flexibility in network planning and service provisioning. In a VTN, the performance of critical services can be further ensured using other mechanisms, e.g. those as defined in [DetNet].
- o Scalability: The introduction of resource aware SIDs for different VTNs would increase the amount of SIDs and state in the network. While the increased network state is considered an inevitable price in meeting the requirements of some customers or services, the SR based VTN mechanism seeks to achieve a balance between the state limitations of traditional end-to-end TE mechanism and the lack of resource awareness in classic segment routing. Following the segment routing paradigm, network resources are allocated on network segments in a per VTN manner and represented as SIDs, this ensures that there is no per-path state introduced in the network. In addition, operators can choose the granularity of resource allocation on different network segments. In network segments where resource is scarce such that the service requirement may not always be met, the proposed approach can be used to allocate a set of resources to a VTN which contains such network segment to avoid possible competition. By contrast, in other segment of the

network where resource is considered plentiful, the resource may be shared between a number of VTNs. The decision to do this is in the hands of the operator. Because of the segmented nature of the SR based VTN, resource aggregation is easier and more flexible than RSVP-TE based approach.

5. Service Assurance of VTN

In order to provide assurance for services provisioned in the SR based VTNs, it is necessary to instrument the network at multiple levels, e.g. in both the underlay network level and the VTN level. The operator or the customer may also monitor and measure the performance of the services carried by the VTN. In principle these can be achieved using existing or in development techniques in IETF. The detailed mechanisms are out of the scope of this document.

In case of failure or service performance degradation happens in a VTN, it is necessary that some recovery mechanisms, e.g. local protection or end-to-end protection mechanism is used to switch the traffic to another path in the same VTN which could meet the service performance requirement. Care must be taken that the service or path recovery mechanism in one VTN does not impact other VTNs in the same network.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

The security considerations of segment routing and resource-aware SIDs are applicable to this document.

The SR VTNs may be used carry services with specific SLA parameters. An attack can be directly targeted at the customer application by disrupting the SLA, and can be targeted at the network operator by causing them to violate their SLA, triggering commercial consequences. By rigorously policing ingress traffic and carefully provisioning the resources provided to the VTN, this type of attack can be prevented. However care needs to be taken when shared resources are provided between VTNs at some point in the network, and when the network needs to be reconfigured as part of ongoing maintenance or in response to a failure.

The details of the underlying network should not be exposed to third parties, some abstraction would be needed, this is also to prevent attacks aimed at exploiting a shared resource between VTNs.

8. Contributors

Zhenbin Li
Email: lizhenbin@huawei.com

Zhibo Hu
Email: huzhibo@huawei.com

9. Acknowledgements

The authors would like to thank Mach Chen, Stefano Previdi, Charlie Perkins, Bruno Decraene, Loa Andersson, Alexander Vainshtein, Joel Halpern, James Guichard and Adrian Farrel for the valuable discussion and suggestions to this document.

10. References

10.1. Normative References

[RFC8402] Filtsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

[RFC8660] Bashandy, A., Ed., Filtsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <<https://www.rfc-editor.org/info/rfc8660>>.

10.2. Informative References

[DetNet] "DetNet WG", 2016, <<https://datatracker.ietf.org/wg/detnet>>.

[I-D.dong-idr-bgppls-sr-enhanced-vpn]
Dong, J., Hu, Z., Li, Z., Tang, X., and R. Pang, "BGP-LS Extensions for Segment Routing based Enhanced VPN", draft-dong-idr-bgppls-sr-enhanced-vpn-02 (work in progress), June 2020.

- [I-D.dong-lsr-sr-enhanced-vpn]
Dong, J., Hu, Z., Li, Z., Tang, X., Pang, R., JooHeon, L., and S. Bryant, "IGP Extensions for Segment Routing based Enhanced VPN", draft-dong-lsr-sr-enhanced-vpn-04 (work in progress), June 2020.
- [I-D.ietf-idr-bgppls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgppls-segment-routing-epe-19 (work in progress), May 2019.
- [I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-13 (work in progress), October 2020.
- [I-D.ietf-spring-resource-aware-segments]
Dong, J., Bryant, S., Miyasaka, T., Zhu, Y., Qin, F., Li, Z., and F. Clad, "Introducing Resource Awareness to SR Segments", draft-ietf-spring-resource-aware-segments-00 (work in progress), July 2020.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-09 (work in progress), November 2020.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-28 (work in progress), December 2020.
- [I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Networks (VPN+) Service", draft-ietf-teas-enhanced-vpn-06 (work in progress), July 2020.
- [I-D.xie-idr-bgppls-sr-vtn-mt]
Xie, C., Li, C., Dong, J., and Z. Li, "BGP-LS with Multi-topology for Segment Routing based Virtual Transport Networks", draft-xie-idr-bgppls-sr-vtn-mt-01 (work in progress), July 2020.

- [I-D.xie-lsr-isis-sr-vtn-mt]
Xie, C., Ma, C., Dong, J., and Z. Li, "Using IS-IS Multi-Topology (MT) for Segment Routing based Virtual Transport Network", draft-xie-lsr-isis-sr-vtn-mt-02 (work in progress), October 2020.
- [I-D.zhu-idr-bgppls-sr-vtn-flexalgo]
Zhu, Y., Dong, J., and Z. Hu, "BGP-LS with Flex-Algo for Segment Routing based Virtual Transport Networks", draft-zhu-idr-bgppls-sr-vtn-flexalgo-00 (work in progress), March 2020.
- [I-D.zhu-lsr-isis-sr-vtn-flexalgo]
Zhu, Y., Dong, J., and Z. Hu, "Using Flex-Algo for Segment Routing based VTN", draft-zhu-lsr-isis-sr-vtn-flexalgo-01 (work in progress), September 2020.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <<https://www.rfc-editor.org/info/rfc4915>>.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/info/rfc8570>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.

Authors' Addresses

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Stewart Bryant
Futurewei Technologies

Email: stewart.bryant@gmail.com

Takuya Miyasaka
KDDI Corporation

Email: ta-miyasaka@kddi.com

Yongqing Zhu
China Telecom

Email: zhuyq8@chinatelecom.cn

Fengwei Qin
China Mobile

Email: qinfengwei@chinamobile.com

Zhenqiang Li
China Mobile

Email: li_zhenqiang@hotmail.com

Francois Clad
Cisco Systems

Email: fclad@cisco.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2020

Y. Ueno, Ed.
NTT Communications Corporation
R. Nakamura
The University of Tokyo
T. Kamata
Cisco Systems, Inc.
October 27, 2019

SRv6 Tagging proxy
draft-eden-srv6-tagging-proxy-00

Abstract

This document describes the tagging method of SRv6 proxy. SRv6 proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 29, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. SRv6 Tagging proxy | 3 |
| 3.1. SRv6 pseudocode | 4 |
| 3.1.1. Tagging proxy for inner type IPv4 | 4 |
| 3.1.2. Tagging proxy for inner type IPv6 | 5 |
| 4. Implementation status | 5 |
| 5. Discussion | 6 |
| 6. Acknowledgements | 6 |
| 7. IANA Considerations | 6 |
| 7.1. SRv6 Endpoint Behaviors | 6 |
| 8. Security Considerations | 6 |
| 9. References | 6 |
| 9.1. Normative References | 6 |
| 9.2. Informative References | 7 |
| Authors' Addresses | 7 |

1. Introduction

Segment Routing (SR) is a source routing architecture defined in [RFC8402]. SR uses segment identifiers (SIDs) to identify each entity in an SR network. SR can be applied two types of data plane, MPLS and IPv6. IPv6 based SR is called Segment Routing IPv6 (SRv6) and its header format is defined in [I-D.ietf-6man-segment-routing-header]. As for the SRv6 packets, the SIDs are embedded in packets in the form of a list with the current index of the list, called SegmentsLeft (SL). Packets with Segment Routing Headers (SRHs) are steered through the ordered list of SIDs. Note that the proxy behavior defined in this document can only be applied for SRv6 packets.

Because SR can steer packets through arbitrary SR nodes, SR can be applied to Service Function Chaining (SFC). SFC, defined in [RFC7665], is an architecture that realizes the on-demand instantiation of an ordered set of service functions. Although there are differences in the specific packet steering method, SR defined in [RFC8402] can realize SFC and [I-D.xuclad-spring-sr-service-programming] describes SR proxy behaviors to integrate SR-unaware services to it.

This document describes a new SRv6 proxy, called tagging proxy. The tagging proxy, which is a variant of the dynamic SR proxy, supports

both IPv4 and IPv6 and multiple service chains by one proxy instance without state management.

2. Terminology

This document leverages the terminology proposed in [RFC8402], [I-D.ietf-spring-segment-routing-policy], and [I-D.xuclad-spring-sr-service-programming].

3. SRv6 Tagging proxy

The proxy is a variant of the dynamic proxy defined in [I-D.xuclad-spring-sr-service-programming]. The dynamic proxy caches the outer IPv6 header and SRH before removing it from the incoming traffic. After removal of the outer IPv6 and SRH headers, the dynamic proxy sends the traffic to an associating service and the same headers are re-attached to the traffic returning from the service.

For caching outer headers, the tagging proxy uses arguments of SRv6 SIDs as indexes for cache entries. The arguments are determined by the operator to correspond one-to-one with the service chains, and the process could be automated by the network controllers. Upon receiving a packet whose active segment matches a tagging SR proxy function, the proxy node caches the IPv6 header and SRH. Corresponding cache entry for a packet is indicated by an argument part of the SRv6 SID. Every time a packet arrives, a corresponding cache entry is updated.

The tagging proxy removes the IPv6 header and SRH for sending the inner packet to the SR-unaware service. At that time the tagging proxy treats the index as a "tag", that is embedded into the inner packet. As a field to embed the tag, Type of Service (ToS) is used for IPv4 packets and Traffic Class (TC) is used for IPv6 packets. Note that the argument length of the SID for tagging proxy cannot be greater than 8-bit because of the length of ToS and TC fields.

When the proxy node receives the packet returning from the SR-unaware service, the proxy node pushes the IPv6 header and SRH onto the packet. The headers are retrieved from the cache entry that corresponds to the tag extracted from the ToS or TC field of the packet.

A tagging SR proxy segment is associated with the following mandatory parameters:

- o NH-ADDR: Next hop Ethernet address (only for inner type IPv4 and IPv6)

- o IFACE-OUT: Local interface for sending traffic towards the service
- o IFACE-IN: Local interface receiving the traffic coming back from the service

A tagging SR proxy segment is thus defined for a specific service. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

3.1. SRv6 pseudocode

3.1.1. Tagging proxy for inner type IPv4

Upon receiving an IPv6 packet destined for S, where S is an IPv6 tagging proxy segment for IPv4 traffic, a node N does:

1. IF NH=SRH & SL > 0 & ENH == 4 THEN
2. Cache IPv6 Header and SRH into CACHE[ARG]
3. Remove the (outer) IPv6 header and its extension headers
4. Embed ARG into the ToS field of the (inner) IPv4 header
5. Forward the exposed packet on IFACE-OUT towards NH-ADDR
6. ELSE
7. Drop the packet

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

1. IF CACHE[ToS] THEN
2. Set ToS value to 0
3. Decrement TTL and update checksum of the inner IPv4 header
4. Push the IPv6 header and SRH in CACHE[ToS]
5. Set ENH value to 4
6. Update the payload length of the outer IPv6 header
7. Lookup outer DA in appropriate table and proceed accordingly
8. ELSE
9. Drop the packet

Note that the proxy may cache and restore the ToS value of inner IPv4 packet in addition to outer IPv6 header and SRH if the service chain uses single ToS value.

3.1.2. Tagging proxy for inner type IPv6

Upon receiving an IPv6 packet destined for S, where S is an IPv6 tagging proxy segment for IPv6 traffic, a node N does:

1. IF NH=SRH & SL > 0 & ENH == 41 THEN
2. Cache IPv6 Header and SRH into CACHE[ARG]
3. Remove the (outer) IPv6 header and its extension headers
4. Embed ARG into the TC field of the (inner) IPv6 header
5. Forward the exposed packet on IFACE-OUT towards NH-ADDR
6. ELSE
7. Drop the packet

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

1. IF CACHE[TC] THEN
2. Set TC value to 0
3. Decrement Hop Limit of the inner IPv6 header
4. Push the IPv6 header and SRH in CACHE[TC]
5. Set ENH value to 41
6. Update the payload length of the outer IPv6 header
7. Lookup outer DA in appropriate table and proceed accordingly
8. ELSE
9. Drop the packet

Note that the proxy may cache and restore the TC value of inner IPv6 packet in addition to outer IPv6 header and SRH if the service chain uses single ToS value.

4. Implementation status

This section is to be removed before publishing as an RFC.

The tagging SR proxy is available on the below open-source implementations.

- o Linux XDP based implementation by Yukito Ueno
- o Linux kernel based implementation (out-of-tree) by Ryo Nakamura

Also, both implementations were operated for the traffic of exhibitors and visitors at Interop Tokyo 2019 ShowNet.

5. Discussion

This tagging proxy uses ToS or Traffic Class field as a container of an index of a cache entry. Upon receiving a packet returning from an SR-unaware service, the index is needed for the proxy node to decide which cache entry should be pushed to the packet. On the other hand, the usage is different from the original purpose of ToS and TC fields.

6. Acknowledgements

The authors would like to thank all the members and contributors of Interop Tokyo 2019 ShowNet. The authors are also thankful to Francois Clad for his comments.

7. IANA Considerations

7.1. SRv6 Endpoint Behaviors

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Behaviors" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry, the following allocations:

| Value | Description | Reference |
|-------|------------------------|-----------|
| TBA | End.AT - Tagging proxy | [This.ID] |

8. Security Considerations

The security requirements and mechanisms described in [RFC8402], [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming] also apply to this document. This document does not introduce any new security vulnerabilities.

9. References

9.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment
Routing Header (SRH)", draft-ietf-6man-segment-routing-
header-26 (work in progress), October 2019.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d.,
bogdanov@google.com, b., and P. Mattes, "Segment Routing
Policy Architecture", draft-ietf-spring-segment-routing-
policy-03 (work in progress), May 2019.
- [I-D.xuclad-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca,
d., Li, C., Decraene, B., Ma, S., Yadlapalli, C.,
Henderickx, W., and S. Salsano, "Service Programming with
Segment Routing", draft-xuclad-spring-sr-service-
programming-02 (work in progress), April 2019.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

9.2. Informative References

- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015,
<<https://www.rfc-editor.org/info/rfc7665>>.

Authors' Addresses

Yukito Ueno (editor)
NTT Communications Corporation
Tokyo
JP

Phone: +80 90 3085 5274
Email: yukito.ueno@ntt.com

Ryo Nakamura
The University of Tokyo
Tokyo
JP

Phone: +81 3 5841 2710
Email: upa@haeena.net

Teppei Kamata
Cisco Systems, Inc.
Tokyo
JP

Email: tkamata@cisco.comt

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 22, 2021

R. Gandhi, Ed.
Z. Ali
C. Filsfils
F. Brockners
Cisco Systems, Inc.
B. Wen
V. Kozak
Comcast
February 18, 2021

MPLS Data Plane Encapsulation for In-situ OAM Data
draft-gandhi-mpls-ioam-sr-06

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the data packet while the packet traverses a path between two nodes in the network. This document defines how IOAM data fields are transported with MPLS data plane encapsulation using new Generic Associated Channel (G-ACh).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Conventions | 3 |
| 2.1. Requirement Language | 3 |
| 2.2. Abbreviations | 3 |
| 3. MPLS Extensions for IOAM Data Fields | 4 |
| 3.1. IOAM Generic Associated Channel | 4 |
| 3.2. IOAM Indicator Labels | 5 |
| 4. Edge-to-Edge IOAM | 5 |
| 4.1. Edge-to-Edge IOAM Indicator Label | 5 |
| 4.2. Procedure for Edge-to-Edge IOAM | 6 |
| 4.3. Edge-to-Edge IOAM Indicator Label Allocation | 7 |
| 5. Hop-by-Hop IOAM | 7 |
| 5.1. Hop-by-Hop IOAM Indicator Label | 7 |
| 5.2. Procedure for Hop-by-Hop IOAM | 8 |
| 5.3. Hop-by-Hop IOAM Indicator Label Allocation | 8 |
| 6. Considerations for IOAM Indicator Label | 9 |
| 6.1. Considerations for ECMP | 9 |
| 6.2. Node Capability | 9 |
| 6.3. MSD Considerations | 9 |
| 6.4. Nested MPLS Encapsulation | 10 |
| 7. MPLS Encapsulation with Control Word and Another G-ACh for IOAM Data Fields | 10 |
| 8. Example MPLS Encapsulations | 12 |
| 8.1. Example SR-MPLS Encapsulation with IOAM | 12 |
| 9. Security Considerations | 13 |
| 10. IANA Considerations | 13 |
| 11. References | 14 |
| 11.1. Normative References | 14 |
| 11.2. Informative References | 15 |
| Acknowledgements | 16 |
| Contributors | 16 |
| Authors' Addresses | 16 |

1. Introduction

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the IOAM data fields are added to the data packets rather than being sent within the probe packets specifically

dedicated to OAM or Performance Measurement (PM). The IOAM data fields are defined in [I-D.ietf-ippm-ioam-data], and can be used for various use-cases for OAM and PM. The IOAM data fields are further updated in [I-D.ietf-ippm-ioam-direct-export] for direct export use-cases and in [I-D.ietf-ippm-ioam-flags] for Loopback and Active flags.

This document defines how IOAM data fields are transported with MPLS data plane encapsulations using new Generic Associated Channel (G-ACh).

2. Conventions

2.1. Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

Abbreviations used in this document:

| | |
|-------|---|
| ECMP | Equal Cost Multi-Path |
| E2E | Edge-To-Edge |
| G-ACh | Generic Associated Channel |
| HbH | Hop-by-Hop |
| IOAM | In-situ Operations, Administration, and Maintenance |
| MPLS | Multiprotocol Label Switching |
| OAM | Operations, Administration, and Maintenance |
| PM | Performance Measurement |
| POT | Proof-of-Transit |
| PSID | Path Segment Identifier |
| PW | PseudoWire |
| SR | Segment Routing |

SR-MPLS Segment Routing with MPLS Data plane

3. MPLS Extensions for IOAM Data Fields

3.1. IOAM Generic Associated Channel

The IOAM data fields are defined in [I-D.ietf-ippm-ioam-data]. The IOAM data fields are carried in the MPLS header as shown in Figure 1. More than one trace options can be present in the IOAM data fields. G-ACh [RFC5586] provides a mechanism to transport OAM and other control messages over MPLS data plane. The IOAM G-ACh header [RFC5586] with new IOAM G-ACh type is added immediately after the MPLS label stack in the MPLS header as shown in Figure 1, before the IOAM data fields.

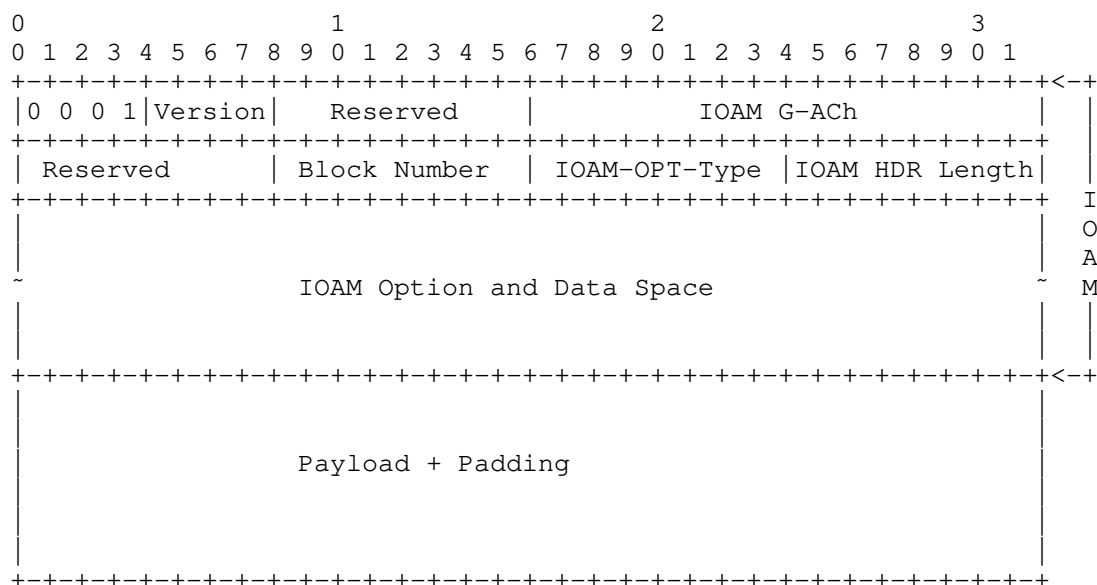


Figure 1: IOAM Generic Associated Channel with IOAM Data Fields

The IOAM data fields are encapsulated using the following fields in the MPLS header:

IP Version Number 0001b: The first four octets are IP Version Field part of a G-ACh header, as defined in [RFC5586].

Version: The Version field is set to 0, as defined in [RFC4385].

IOAM G-ACh: Generic Associated Channel (G-ACh) Type (value TBA3) for IOAM [RFC5586].

Reserved: Reserved Bits MUST be set to zero upon transmission and ignored upon receipt.

Block Number: The Block Number can be used to aggregate the IOAM data collected in data plane, e.g. compute measurement metrics for each block of a flow. It is also used to correlate the IOAM data on different nodes.

IOAM-OPT-Type: 8-bit field defining the IOAM Option type, as defined in Section 8.1 of [I-D.ietf-ippm-ioam-data].

IOAM HDR LEN: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

IOAM Option and Data Space: IOAM option header and data is present as defined by the IOAM-OPT-Type field, and is defined in Section 5 of [I-D.ietf-ippm-ioam-data].

3.2. IOAM Indicator Labels

An IOAM Indicator Label is used to indicate the presence of the IOAM data fields in the MPLS header. There are two IOAM types defined in this document: Edge-to-Edge (E2E) and Hop-by-Hop (HbH) IOAM. If only edge nodes need to process IOAM data then E2E IOAM Indicator Label is used so that intermediate nodes can ignore it. If both edge and intermediate nodes need to process IOAM data then HbH IOAM Indicator Label is used. Different IOAM Indicator Labels allow to optimize the IOAM processing on intermediate nodes by checking if IOAM data fields need to be processed.

4. Edge-to-Edge IOAM

4.1. Edge-to-Edge IOAM Indicator Label

The E2E IOAM Indicator Label is used to indicate the presence of the E2E IOAM data fields in the MPLS header as shown in Figure 2.

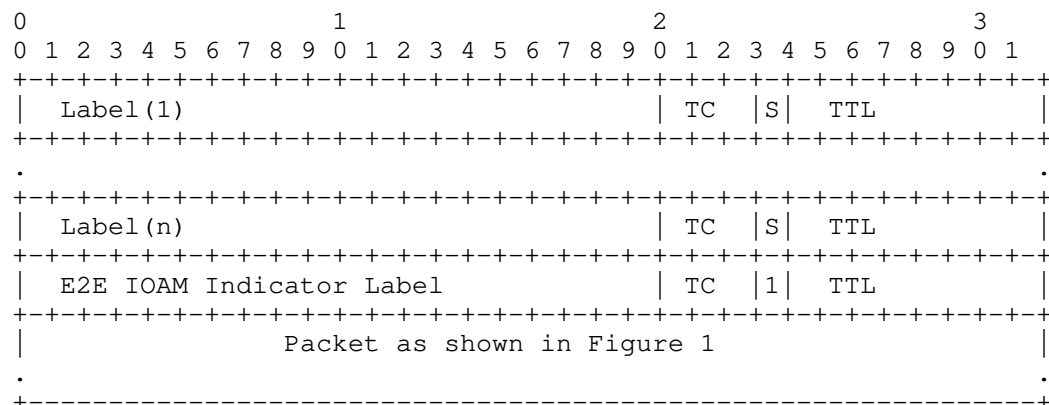


Figure 2: MPLS Encapsulation for E2E IOAM

The E2E IOAM data fields carry the Option-Type(s) that require processing on the encapsulating and decapsulating nodes only. The IOAM Option-Type carried can be IOAM Edge-to-Edge Option-Type [I-D.ietf-ippm-ioam-data]. The E2E IOAM data fields SHOULD NOT carry any IOAM Option-Type that require IOAM processing on the intermediate nodes as it will not be processed by them.

4.2. Procedure for Edge-to-Edge IOAM

The E2E IOM procedure is summarized as following:

- o The encapsulating node inserts the E2E IOAM Indicator Label and one or more IOAM data fields in the MPLS header.
- o The intermediate nodes do not process IOAM data fields.
- o The decapsulating node "punts the timestamped copy" of the received packet as is including the IOAM data fields when the node recognizes the IOAM Indicator Label. The copy of the packet is punted with receive timestamp to the slow path for IOAM data fields processing. The receive timestamp is required by the various E2E OAM use-cases, including streaming telemetry. Note that it is not necessarily punted to the control-plane.
- o The decapsulating node processes the IOAM data fields using the procedures defined in [I-D.ietf-ippm-ioam-data]. An example of IOAM processing is to export the data fields, send data fields via streaming telemetry, etc.
- o The decapsulating node also pops the IOAM Indicator Label and the IOAM data fields from the received packet. The decapsulated

packet is forwarded downstream or terminated locally similar to the regular data packets.

4.3. Edge-to-Edge IOAM Indicator Label Allocation

The E2E IOAM Indicator Label is used to indicate the presence of the E2E IOAM data fields in the MPLS header. The E2E IOAM Indicator Label can be allocated using one of the following three methods:

- o Label assigned by IANA with value TBA1 from the Extended Special-Purpose MPLS Values [I-D.ietf-mpls-spl-terminology].
- o Label allocated by a Controller from the global table of the decapsulating node. The Controller provisions the label on both encapsulating and decapsulating nodes.
- o Label allocated by the decapsulating node and signalled or advertised in the network. The signaling and/or advertisement extension for this is outside the scope of this document.

5. Hop-by-Hop IOAM

5.1. Hop-by-Hop IOAM Indicator Label

The HbH IOAM Indicator Label is used to indicate the presence of the HbH IOAM data fields in the MPLS header as shown in Figure 3.

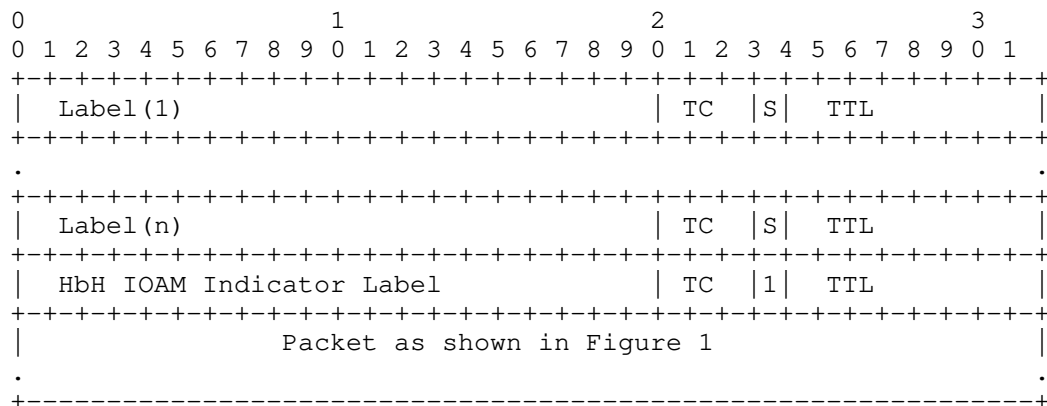


Figure 3: MPLS Encapsulation for HbH IOAM

The HbH IOAM data fields carry the Option-Type(s) that require processing at the intermediate and/or encapsulating and decapsulating nodes. The IOAM Option-Type carried can be IOAM Pre-allocated Trace Option-Type, IOAM Incremental Trace Option-Type and IOAM Proof of

Transit (POT) Option-Type, as well as Edge-to-Edge Option-Type [I-D.ietf-ippm-ioam-data].

5.2. Procedure for Hop-by-Hop IOAM

The HbH IOAM procedure is summarized as following:

- o The encapsulating node inserts the HbH IOAM Indicator Label and one or more IOAM data fields in the MPLS header.
- o The intermediate node enabled with HbH IOAM functions processes the data packet including the IOAM data fields as defined in [I-D.ietf-ippm-ioam-data] when the node recognizes the HbH IOAM Indicator Label present in the MPLS header. The intermediate node may 'punt the timestamped copy' of the received data packet including the IOAM data fields as required by the IOAM data fields processing. The copy of the packet is punted with receive timestamp to the slow path for IOAM processing.
- o The intermediate node forwards a copy of the processed data packet downstream.
- o The decapsulating node "punts the timestamped copy" of the received data packet as is including the IOAM data fields when the node recognizes the IOAM Indicator Label. The copy of the packet is punted with receive timestamp to the slow path for IOAM data fields processing. The receive timestamp is required by the various E2E OAM use-cases, including streaming telemetry. Note that it is not necessarily punted to the control-plane.
- o The decapsulating node processes the IOAM data fields using the procedures defined in [I-D.ietf-ippm-ioam-data]. An example of IOAM processing is to export the data fields, send data fields via streaming telemetry, etc.
- o The decapsulating node also pops the IOAM Indicator Label and the IOAM data fields from the received packet. The decapsulated packet is forwarded downstream or terminated locally similar to the regular data packets.

5.3. Hop-by-Hop IOAM Indicator Label Allocation

The HbH IOAM Indicator Label is used to indicate the presence of the HbH IOAM data fields in the MPLS header. The HbH IOAM Indicator Label can be allocated using one of the following three methods:

- o Label assigned by IANA with value TBA2 from the Extended Special-Purpose MPLS Values [I-D.ietf-mpls-spl-terminology].

- o Label allocated by a Controller from the network-wide global table. The Controller provisions the labels on all nodes participating in IOAM functions along the data traffic path.
- o Labels allocated by the intermediate and decapsulating nodes and signalled or advertised in the network. The signaling and/or advertisement extension for this is outside the scope of this document.

6. Considerations for IOAM Indicator Label

6.1. Considerations for ECMP

The encapsulating node needs to make sure the IOAM data fields do not start with a well-known IP Version Number (e.g. 0x4 for IPv4 and 0x6 for IPv6) as that can alter the hashing function for ECMP that uses the IP header. This is achieved by using the IOAM G-ACh with IP Version Number 0001b after the MPLS label stack [RFC5586].

Note that the hashing function for ECMP that uses the labels from the MPLS header may now include the IOAM Indicator Label.

When entropy label [RFC6790] is used for hashing function for ECMP, the procedure defined in this document does not alter the hashing function.

6.2. Node Capability

The decapsulating node that has to pop the IOAM Indicator Label, data fields, and perform the IOAM function may not be capable of supporting it. The encapsulating node needs to know if the decapsulating node can support the IOAM function. The signaling extension for this capability exchange is outside the scope of this document.

The intermediate node that is not capable of supporting the IOAM functions defined in this document, can simply skip the IOAM processing of the MPLS header.

6.3. MSD Considerations

The SR path computation needs to know the Maximum SID Depth (MSD) that can be imposed at each node/link of a given SR path [RFC8664]. This ensures that the SID stack depth of a computed path does not exceed the number of SIDs the node is capable of imposing. The MSD used for path computation MUST include the IOAM Indicator Label.

6.4. Nested MPLS Encapsulation

The data packets with IOAM data fields carry only one IOAM Indicator Label in the MPLS header. Any intermediate node that adds additional MPLS encapsulation in the MPLS header may further update the IOAM data fields in the header without inserting another IOAM Indicator Label. When a packet is received with a HbH IOAM Indicator Label, the nested MPLS encapsulating node can add a HbH and/or E2E IOAM Option-Type. However, when a packet is received with an E2E IOAM Indicator Label, the nested MPLS encapsulating node SHOULD NOT add a HbH IOAM Option-Type, as intermediate nodes will not process it.

7. MPLS Encapsulation with Control Word and Another G-ACh for IOAM Data Fields

The IOAM data fields, including IOAM G-ACh header are added in the MPLS encapsulation immediately after the MPLS header. Any Control Word [RFC4385] or another G-ACh [RFC5586] MUST be added after the IOAM data fields in the packet as shown in the Figure 4 and Figure 5, respectively. This allows the intermediate nodes to easily access the HbH IOAM data fields located immediately after the MPLS header. The decapsulating node can remove the MPLS encapsulation including the IOAM data fields and then process the Control Word or another G-ACh following it.

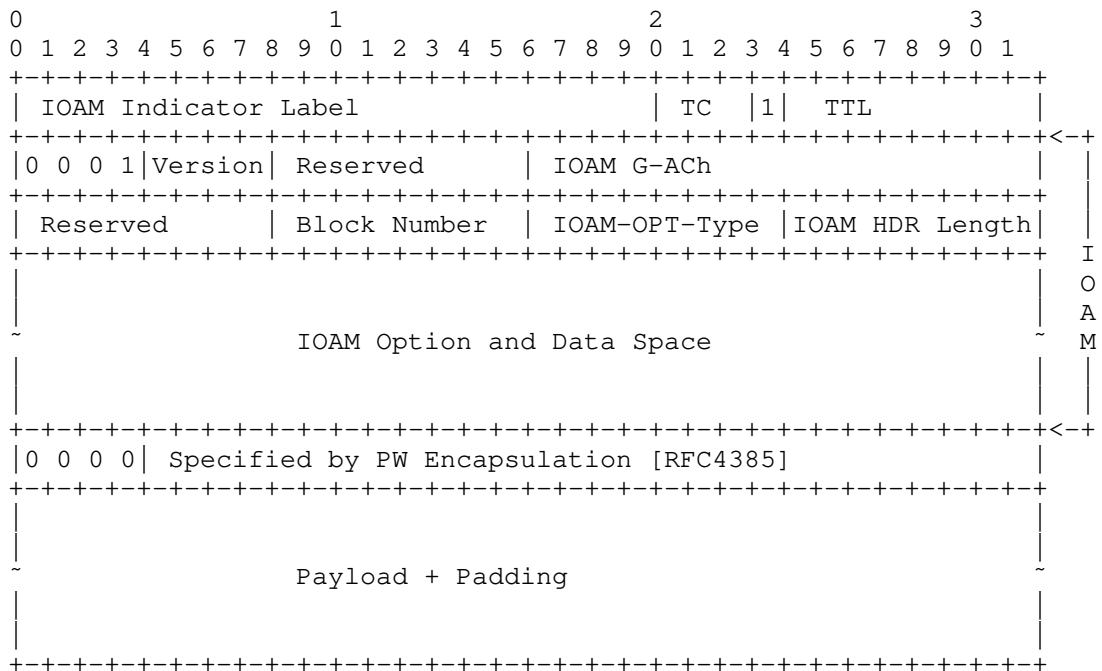


Figure 4: Example MPLS Encapsulation with Generic PW Control Word with IOAM

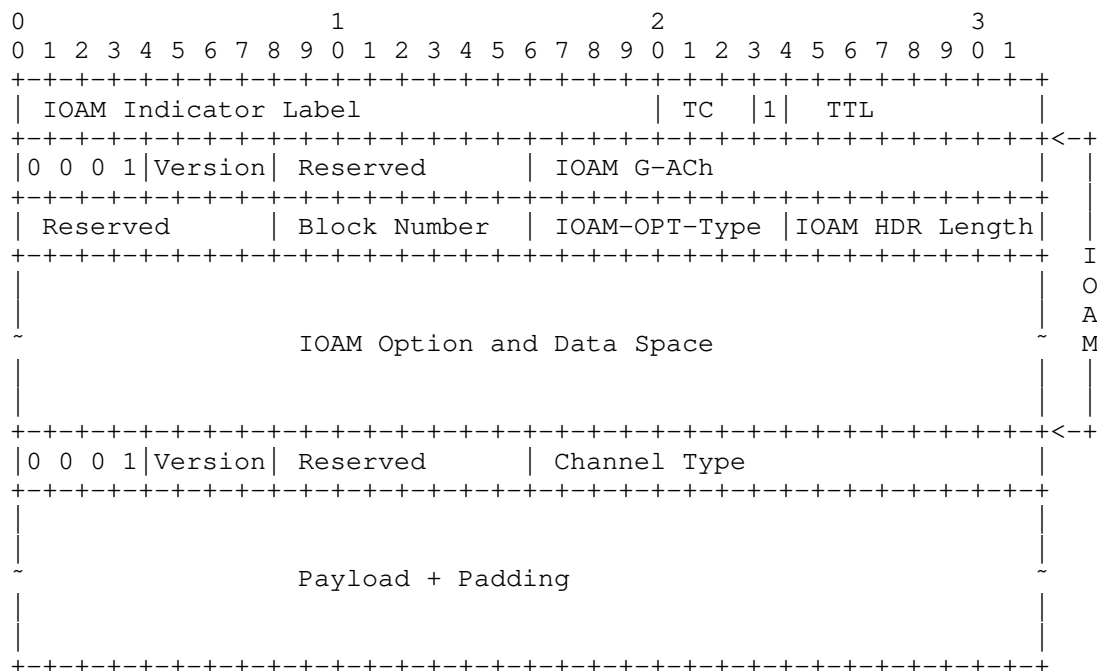


Figure 5: Example MPLS Encapsulation with Another G-ACh with IOAM

8. Example MPLS Encapsulations

8.1. Example SR-MPLS Encapsulation with IOAM

Segment Routing (SR) technology leverages the source routing paradigm [RFC8660]. A node steers a packet through a controlled set of instructions, called segments, by pre-pending the packet with an SR header. In the SR with MPLS data plane (SR-MPLS), the SR header is instantiated through a label stack.

An example of data packet with SR-MPLS encapsulation containing Path Segment Identifier (PSID) [I-D.ietf-spring-mpls-path-segment] and E2E IOAM data fields is shown in Figure 6. The PSID allows to identify the path associated with the data traffic being monitored for IOAM on the decapsulating node.

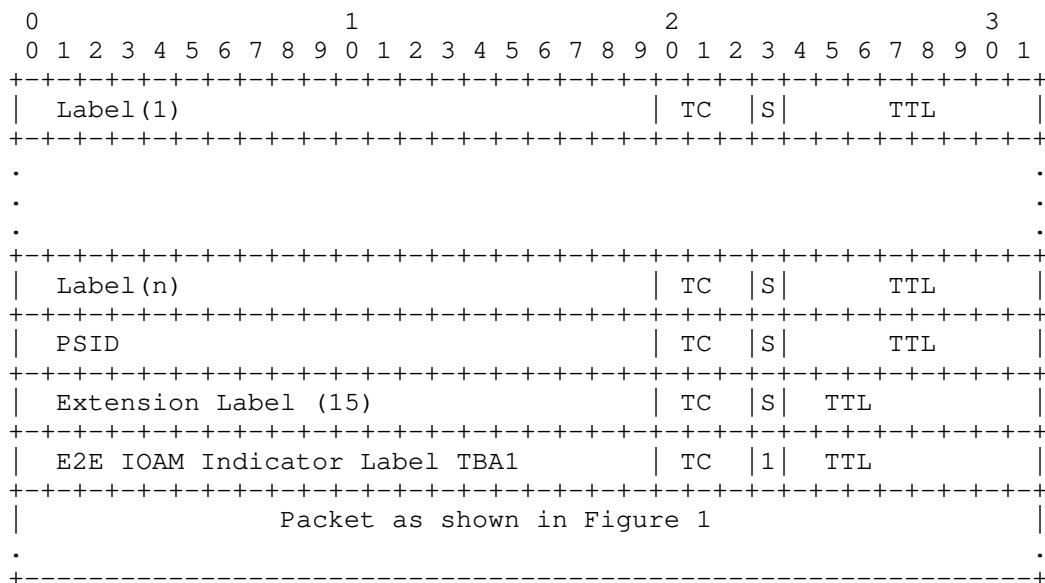


Figure 6: Example SR-MPLS Encapsulation with E2E IOAM Data Fields

9. Security Considerations

The security considerations of IOAM in general are discussed in [I-D.ietf-ippm-ioam-data].

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain.

Routers that support G-ACh are subject to the same security considerations as defined in [RFC4385] and [RFC5586].

10. IANA Considerations

IANA maintains the "Special-Purpose Multiprotocol Label Switching (MPLS) Label Values" registry (see <<https://www.iana.org/assignments/mpls-label-values/mpls-label-values.xml>>). IANA is requested to allocate IOAM Indicator Label value from the "Extended Special-Purpose MPLS Label Values" registry:

| Value | Description | Reference |
|-------|--------------------------|---------------|
| TBA1 | E2E IOAM Indicator Label | This document |
| TBA2 | HbH IOAM Indicator Label | This document |

Table 1: IOAM Indicator Label Values

IANA maintains G-ACh Type Registry (see <https://www.iana.org/assignments/g-ach-parameters/g-ach-parameters.xhtml>). IANA is requested to allocate a value for IOAM G-ACh Type from "MPLS Generalized Associated Channel (G-ACh) Types (including Pseudowire Associated Channel Types)" registry.

| Value | Description | Reference |
|-------|-----------------|---------------|
| TBA3 | IOAM G-ACh Type | This document |

Table 2: IOAM G-ACh Type

11. References

11.1. Normative References

- [I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-11 (work in progress), November 2020.
- [I-D.ietf-ippm-ioam-direct-export]
Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F., Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ OAM Direct Exporting", draft-ietf-ippm-ioam-direct-export-02 (work in progress), November 2020.
- [I-D.ietf-ippm-ioam-flags]
Mizrahi, T., Brockners, F., Bhandari, S., Sivakolundu, R., Pignataro, C., Kfir, A., Gafni, B., Spiegel, M., and J. Lemon, "In-situ OAM Flags", draft-ietf-ippm-ioam-flags-03 (work in progress), October 2020.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [I-D.ietf-mpls-spl-terminology] Andersson, L., Kompella, K., and A. Farrel, "Special Purpose Label terminology", draft-ietf-mpls-spl-terminology-06 (work in progress), January 2021.
- [I-D.ietf-spring-mpls-path-segment] Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler, "Path Segment in MPLS Based Segment Routing Network", draft-ietf-spring-mpls-path-segment-03 (work in progress), September 2020.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <<https://www.rfc-editor.org/info/rfc8660>>.
- [RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <<https://www.rfc-editor.org/info/rfc8664>>.

Acknowledgements

The authors would like to thank Patrick Khordoc, Shwetha Bhandari and Vengada Prasad Govindan for the discussions on IOAM. The authors would also like to thank Tarek Saad, Loa Andersson, Greg Mirsky, Stewart Bryant, Xiao Min, and Cheng Li for providing many useful comments. The authors would also like to thank Mach Chen, Andrew Malis, Matthew Bocci, and Nick Delregno for the MPLS-RT reviews.

Contributors

Sagar Soni
Cisco Systems, Inc.

Email: sagsoni@cisco.com

Authors' Addresses

Rakesh Gandhi (editor)
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

Voitek Kozak
Comcast

Email: Voitek_Kozak@comcast.com

SPRING Working Group
Internet-Draft
Intended Status: Standards Track
Expires: May 7, 2020

R. Gandhi, Ed.
C. Filsfils
Cisco Systems, Inc.
D. Voyer
Bell Canada
M. Chen
Huawei
B. Janssens
Colt
November 4, 2019

Performance Measurement Using TWAMP Light
for Segment Routing Networks
draft-gandhi-spring-twamp-srpm-04

Abstract

Segment Routing (SR) leverages the source routing paradigm. SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. This document specifies procedure for sending and processing probe query and response messages for Performance Measurement (PM) in Segment Routing networks. The procedure uses the mechanisms defined in RFC 5357 (Two-Way Active Measurement Protocol (TWAMP) Light) and Simple Two-Way Active Measurement Protocol (STAMP) for Delay Measurement, and also uses the mechanisms defined in this document for Loss Measurement. The procedure specified is applicable to SR-MPLS and SRv6 data planes and are used for both links and end-to-end SR Policies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Conventions Used in This Document | 4 |
| 2.1. Requirements Language | 4 |
| 2.2. Abbreviations | 4 |
| 2.3. Reference Topology | 5 |
| 3. Overview | 5 |
| 3.1. Example Provisioning Model | 6 |
| 3.2. STAMP Applicability | 7 |
| 4. Probe Messages | 7 |
| 4.1. Probe Query Message | 7 |
| 4.1.1. Delay Measurement Probe Query Message | 7 |
| 4.1.1.1. Delay Measurement Authentication Mode | 8 |
| 4.1.2. Loss Measurement Probe Query Message | 8 |
| 4.1.2.1. Loss Measurement Authentication Mode | 12 |
| 4.1.3. Probe Query for SR Links | 13 |
| 4.1.4. Probe Query for End-to-end Measurement for SR Policy | 13 |
| 4.1.4.1. Probe Query Message for SR-MPLS Policy | 13 |
| 4.1.4.2. Probe Query Message for SRv6 Policy | 13 |
| 4.2. Probe Response Message | 14 |
| 4.2.1. One-way Measurement Mode | 18 |
| 4.2.2. Two-way Measurement Mode | 18 |
| 4.2.2.1. Return Path TLV | 18 |
| 4.2.2.2. Probe Response Message for SR-MPLS Policy | 20 |
| 4.2.2.3. Probe Response Message for SRv6 Policy | 20 |
| 4.2.3. Loopback Measurement Mode | 21 |
| 5. Performance Measurement for P2MP SR Policies | 21 |
| 6. ECMP Support for SR Policies | 22 |
| 7. Additional Message Processing Rules | 22 |
| 7.1. TTL Value | 22 |
| 7.2. Router Alert Option | 22 |
| 7.3. UDP Checksum | 23 |

| | |
|--|----|
| 8. Security Considerations | 23 |
| 9. IANA Considerations | 23 |
| 10. References | 24 |
| 10.1. Normative References | 24 |
| 10.2. Informative References | 25 |
| Acknowledgments | 27 |
| Authors' Addresses | 27 |

1. Introduction

Segment Routing (SR) leverages the source routing paradigm and greatly simplifies network operations for Software Defined Networks (SDNs). SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. SR takes advantage of the Equal-Cost Multipaths (ECMPs) between source and transit nodes, between transit nodes and between transit and destination nodes. SR Policies as defined in [I-D.spring-segment-routing-policy] are used to steer traffic through a specific, user-defined paths using a stack of Segments. Built-in SR Performance Measurement (PM) is one of the essential requirements to provide Service Level Agreements (SLAs).

The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks using probe messages. These protocols rely on control-channel signaling to establish a test-channel over an UDP path. These protocols lack support for direct-mode Loss Measurement (LM) to detect actual data traffic loss which is required in SR networks. The Simple Two-way Active Measurement Protocol (STAMP) [I-D.ippm-stamp] alleviates the control-channel signaling by using configuration data model to provision a test-channel. The TWAMP Light [Appendix I in RFC5357] [BBF.TR-390] provides simplified mechanisms for active performance measurement in Customer IP networks by provisioning UDP paths and eliminates the control-channel signaling.

This document specifies procedures for sending and processing probe query and response messages for Performance Measurement in SR networks. The procedure uses the mechanisms defined in [RFC5357] (TWAMP Light) and STAMP for Delay Measurement (DM), and also uses the mechanisms defined in this document for Loss Measurement. The procedure specified is applicable to SR-MPLS and SRv6 data planes and are used for both links and end-to-end SR Policies. This document also defines mechanisms for handling ECMPs of SR Policies for performance delay measurements.

2. Conventions Used in This Document

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

BSID: Binding Segment ID.

DM: Delay Measurement.

ECMP: Equal Cost Multi-Path.

HMAC: Hashed Message Authentication Code.

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

NTP: Network Time Protocol.

OWAMP: One-Way Active Measurement Protocol.

PM: Performance Measurement.

PSID: Path Segment Identifier.

PTP: Precision Time Protocol.

SID: Segment ID.

SL: Segment List.

SR: Segment Routing.

SRH: Segment Routing Header.

SR-MPLS: Segment Routing with MPLS data plane.

SRv6: Segment Routing with IPv6 data plane.

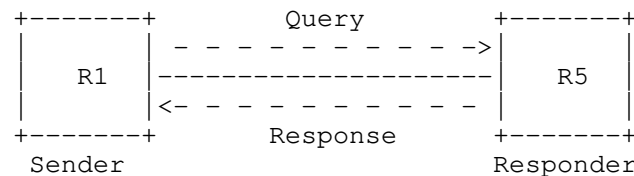
STAMP: Simple Two-way Active Measurement Protocol.

TC: Traffic Class.

TWAMP: Two-Way Active Measurement Protocol.

2.3. Reference Topology

In the reference topology shown below, the sender node R1 initiates a probe query for performance measurement and the responder node R5 sends a probe response for the query message received. The probe response is sent to the sender node R1. The nodes R1 and R5 may be directly connected via a link enabled with Segment Routing or there exists a Point-to-Point (P2P) SR Policy [I-D.spring-segment-routing-policy] on node R1 with destination to node R5. In case of Point-to-Multipoint (P2MP), SR Policy originating from source node R1 may terminate on multiple destination leaf nodes [I-D.spring-sr-p2mp-policy].



Reference Topology

3. Overview

For one-way, two-way and round-trip delay measurements in Segment Routing networks, the TWAMP Light procedures defined in Appendix I of [RFC5357] are used. For one-way and two-way direct-mode and inferred-mode loss measurements in Segment Routing networks, the procedures defined in this document are used. One-way loss measurement provides receive packet loss whereas two-way loss measurement provides both transmit and receive packet loss. Separate UDP destination port numbers are user-configured for delay and loss measurements from the range specified in [I-D.ippm-stamp]. The sender uses the UDP port number following the guidelines specified in Section 6 in [RFC6335]. For both links and end-to-end SR Policies, no PM session for delay or loss measurement is created on the responder node R5 [RFC5357].

For Performance Measurement, probe query and response messages are sent as following:

- o For Delay Measurement, the probe messages are sent on the congruent path of the data traffic by the sender node, and are

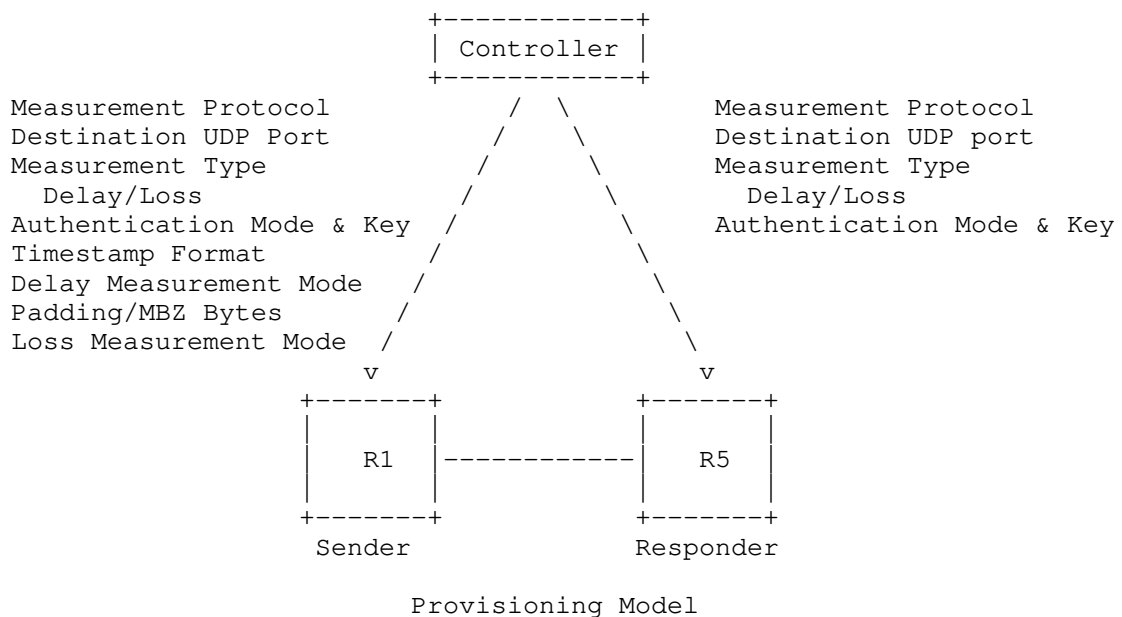
used to measure the delay experienced by the actual data traffic flowing on the links and SR Policies.

- o For Loss Measurement, the probe messages are sent on the congruent path of the data traffic by the sender node, and are used to collect the receive traffic counters for the incoming link or incoming SID where the probe query messages are received at the responder node (incoming link or incoming SID needed since the responder node does not have PM session state present).

The In-Situ Operations, Administration, and Maintenance (IOAM) mechanisms for SR-MPLS defined in [I-D.spring-ioam-sr-mpls] and for SRv6 defined in [I-D.spring-ioam-srv6] are used to carry PM information such as timestamp in-band as part of the data packets, and are outside the scope of this document.

3.1. Example Provisioning Model

An example of a provisioning model and typical measurement parameters for performance delay and loss measurements using TWAMP Light is shown in the following Figure:



The responder node R5 uses the parameters for the timestamp format, delay measurement mode, padding size and loss measurement mode from the received probe query message.

The mechanisms to provision the sender and responder nodes are outside the scope of this document.

3.2. STAMP Applicability

The Simple Two-way Active Measurement Protocol (STAMP) [I-D.ippm-stamp] and the STAMP TLVs [I-D.ippm-stamp-option-tlv] are both equally applicable to the procedures specified in this document.

This is because the delay measurement message formats defined for STAMP and STAMP TLVs are backwards compatible with the delay measurement message formats defined in [RFC5357]. Hence, the same user-configured destination UDP port for delay measurement can be used for STAMP and TWAMP Light messages. The STAMP with a TLV for "direct measurement" can be used for combined delay + loss measurement using a separate user-configured UDP destination port.

The loss measurement probe and query messages defined in this document are also equally applicable to STAMP and STAMP TLVs, and use the message formats defined in [I-D.ippm-stamp].

4. Probe Messages

4.1. Probe Query Message

In this document, the probe messages defined in [RFC5357] are used for Delay and Loss measurements for SR links and end-to-end SR Policies. The user-configured destination UDP ports (separate UDP ports for different delay and loss message formats) are used for identifying the PM probe packets as described in Appendix I of [RFC5357].

4.1.1. Delay Measurement Probe Query Message

The message content for Delay Measurement probe query message using UDP header [RFC768] is shown in Figure 1. The DM probe query message is sent with user-configured Destination UDP port number for DM. The Destination UDP port cannot be used as Source port, since the message does not have any indication to distinguish between query and response. The DM probe query message contains the payload for delay measurement defined in Section 4.1.2 of [RFC5357]. For symmetrical size query and response messages [RFC6038], the DM probe query message contains the payload format defined in Section 4.2.1 of [RFC5357].

```

+-----+
| IP Header                                     |
. Source IP Address = Sender IPv4 or IPv6 Address .

```

```

. Destination IP Address = Responder IPv4 or IPv6 Address      .
. Protocol = UDP                                              .
.                                                            .
+-----+
| UDP Header                                                    |
. Source Port = As chosen by Sender                            .
. Destination Port = User-configured Port for Delay Measurement.
.                                                            .
+-----+
| Payload = Message as specified in Section 4.2.1 of RFC 5357 | |
. Payload = Message as specified in Section 4.1.2 of RFC 5357 | .
. Payload = Message specified in Section 4.2 of [I-D.ippm-stamp].
.                                                            .
+-----+

```

Figure 1: DM Probe Query Message

Timestamp field is eight bytes and use the format defined in Section 4.2.1 of [RFC5357] . It is recommended to use the IEEE 1588v2 Precision Time Protocol (PTP) truncated 64-bit timestamp format [IEEE1588] as specified in [RFC8186].

4.1.1.1. Delay Measurement Authentication Mode

When using the authenticated mode for delay measurement, the matching authentication type (e.g. HMAC-SHA-256) and key are user-configured on both the sender and responder nodes. A separate user-configured destination UDP port is used for the delay measurement in authentication mode due to the different probe message format.

4.1.2. Loss Measurement Probe Query Message

In this document, new probe query message formats are defined for loss measurement as shown in Figure 3A and Figure 3B. The message formats are hardware efficient due to the small size payload and well-known locations of counters. They are similar to the delay measurement message formats and do not require any backwards compatibility and support for the existing DM message formats from [RFC5357].

The message content for Loss Measurement probe query message using UDP header [RFC768] is shown in Figure 2. The LM probe query message is sent with user-configured Destination UDP port number for LM. Separate Destination UDP ports are used for direct-mode and inferred-mode loss measurements. The Destination UDP port cannot be used as Source port, since the message does not have any indication to distinguish between query and response. The LM probe query message contains the payload for loss measurement as defined in

Figure 3A and Figure 3B. For symmetrical size query and response messages [RFC6038], the LM probe query message contains the payload format defined in Figure 7A and Figure 7B for loss measurement.

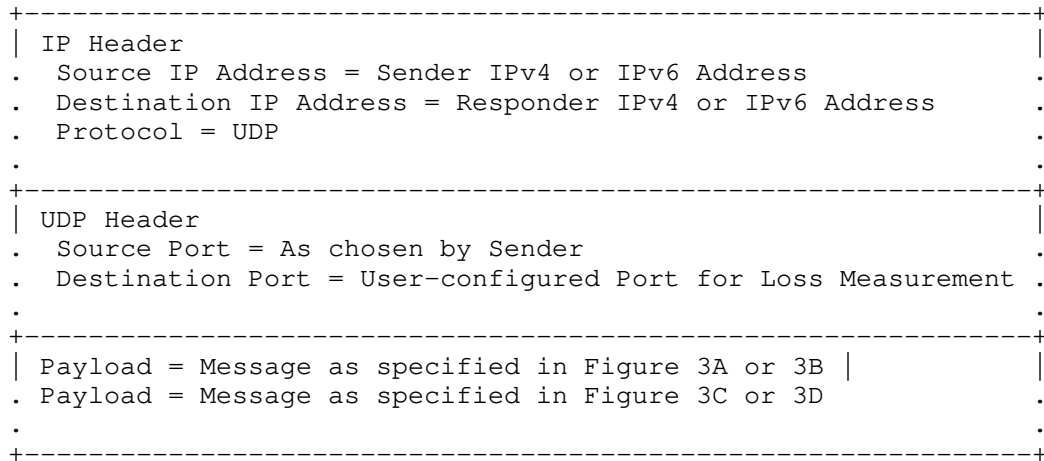


Figure 2: LM Probe Query Message

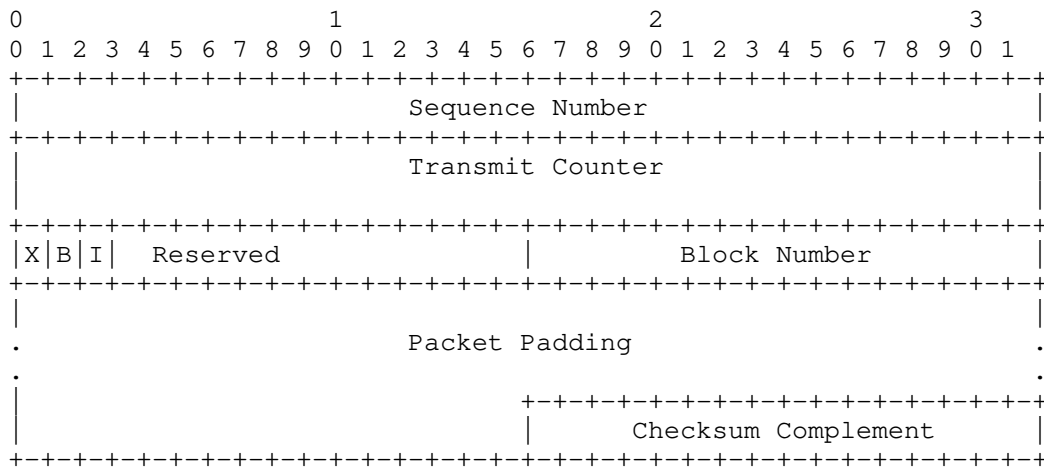
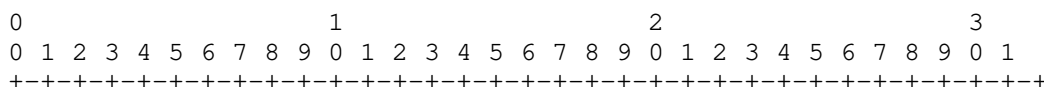


Figure 3A: LM Probe Query Message Format



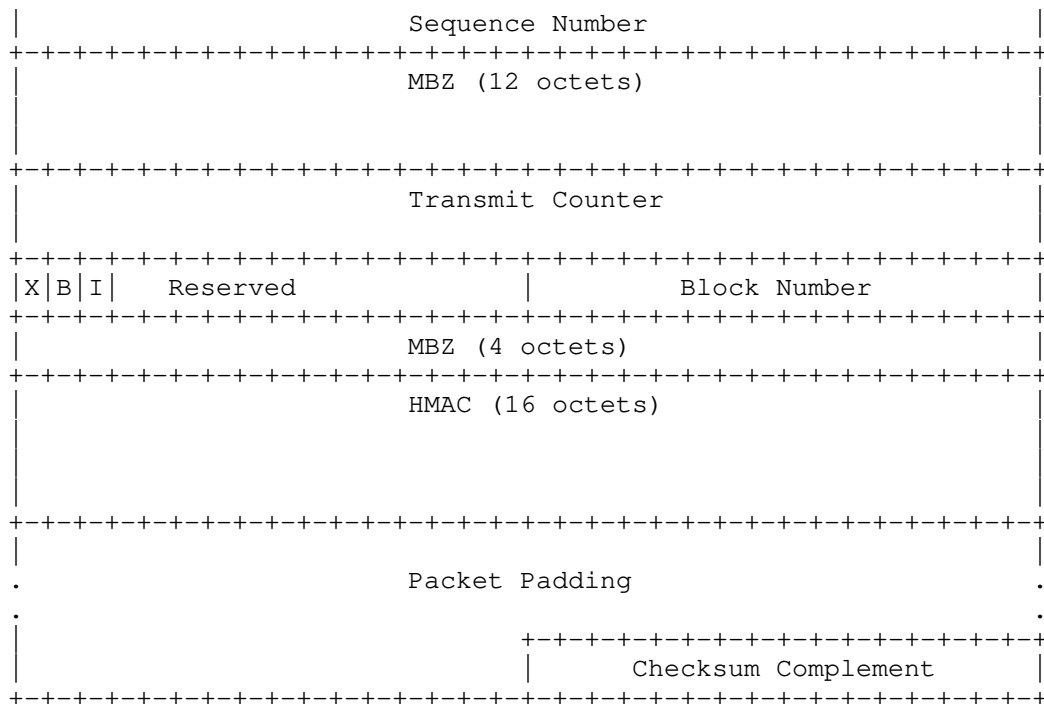


Figure 3B: LM Probe Query Message Format - Authenticated Mode

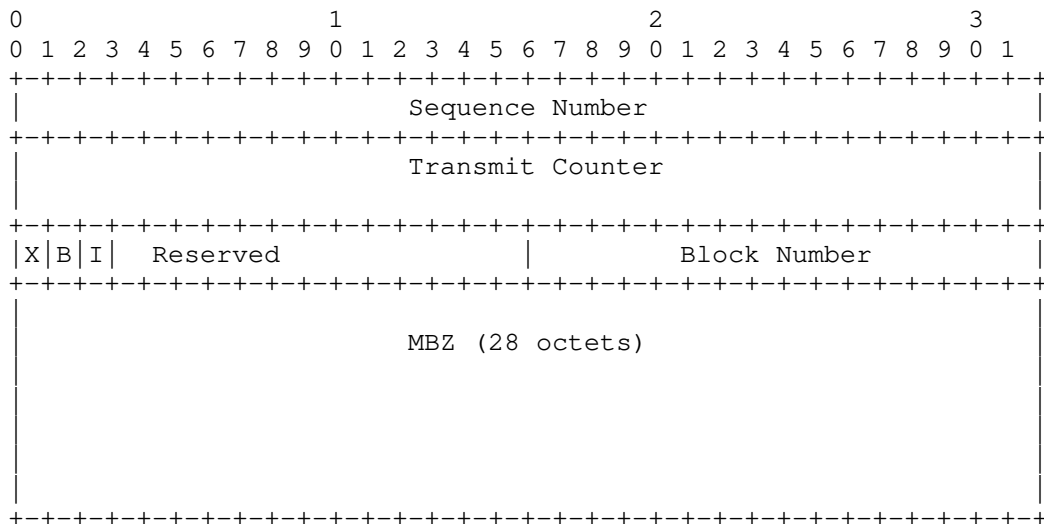


Figure 3C: STAMP LM Probe Query Message Format

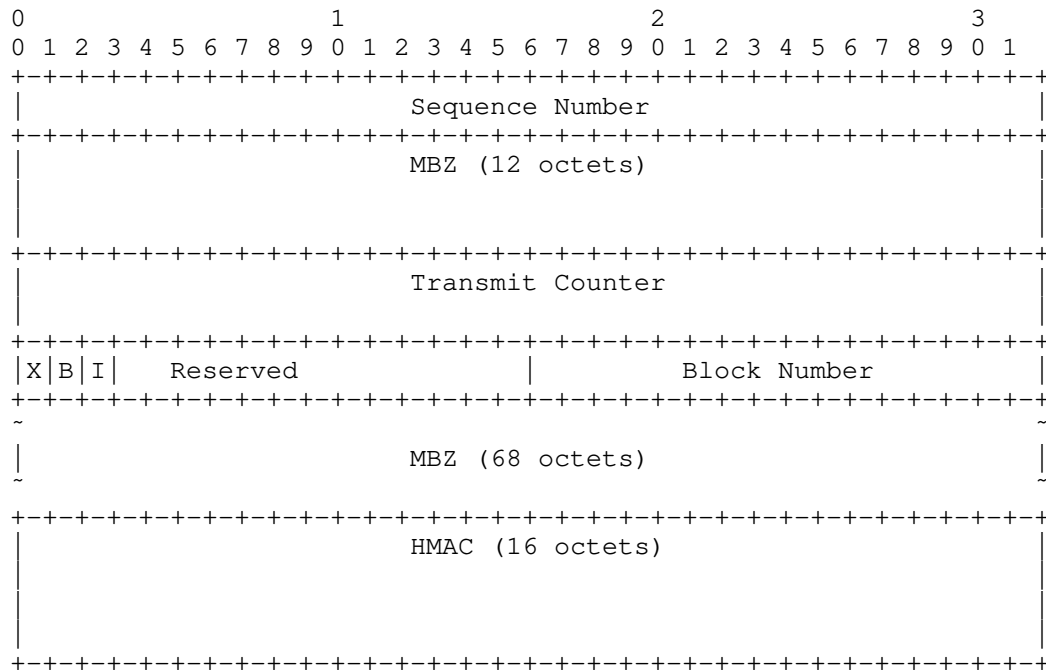


Figure 3D: STAMP LM Probe Query Message Format - Authenticated Mode

Sequence Number (32-bit): As defined in [RFC5357].

Transmit Counter (64-bit): The number of packets sent by the sender node in the query message and by the responder node in the response message. The counter is always written at the fixed location in the probe query and response messages.

Receive Counter (64-bit): The number of packets received at the responder node. It is written by the responder node in the probe response message.

Sender Counter (64-bit): This is the exact copy of the transmit counter from the received query message. It is written by the responder node in the probe response message.

Sender Sequence Number (32-bit): As defined in [RFC5357].

Sender TTL: As defined in [RFC5357].

Flag: The meanings of the Flag bits are:

X: Extended counter format indicator. Indicates the use of extended (64-bit) counter values. Initialized to 1 upon creation (and prior to transmission) of an LM Query and copied from an LM Query to an LM response. Set to 0 when the LM message is transmitted or received over an interface that writes 32-bit counter values.

B: Octet (byte) count. When set to 1, indicates that the Counter 1-4 fields represent octet counts. The octet count applies to all packets within the LM scope, and the octet count of a packet sent or received includes the total length of that packet (but excludes headers, labels, or framing of the channel itself). When set to 0, indicates that the Counter fields represent packet counts.

I: Inferred Mode Loss Measurement: When set to 1, indicates that inferred-mode of loss measurement is used. When set to 0, it indicates direct-mode of loss measurement is used.

Block Number (16-bit): The Loss Measurement using Alternate-Marking method defined in [RFC8321] requires to identify the Block Number (or color) of the traffic counters. The probe query and response messages carry Block Number for the traffic counters for loss measurement. In both probe query and response messages, the counters MUST belong to the same Block Number.

HMAC: The PM probe packet in authenticated mode includes a key Hashed Message Authentication Code (HMAC) ([RFC2104]) hash. Each probe query and response messages are authenticated by adding Sequence Number with Hashed Message Authentication Code (HMAC) TLV. It can use HMAC-SHA-256 truncated to 128 bits (similarly to the use of it in IPSec defined in [RFC4868]); hence the length of the HMAC field is 16 octets.

HMAC uses its own key and the mechanism to distribute the HMAC key is outside the scope of this document.

In authenticated mode, only the sequence number is encrypted, and the other payload fields are sent in clear text. The probe packet MAY include Comp.MBZ (Must Be Zero) variable length field to align the packet on 16 octets boundary.

4.1.2.1. Loss Measurement Authentication Mode

When using the authenticated mode for loss measurement, the matching authentication type (e.g. HMAC-SHA-256) and key are user-configured on both the sender and responder nodes. A separate user-configured destination UDP port is used for the loss measurement in

authentication mode due to the different message format.

4.1.3. Probe Query for SR Links

The probe query message as defined in Figure 1 is sent on the congruent path of the data traffic for Delay measurement. The probe query message as defined in Figure 2 is sent on the congruent path of the data traffic for Loss measurement.

4.1.4. Probe Query for End-to-end Measurement for SR Policy

The performance delay and loss measurement for segment routing is applicable to both SR-MPLS and SRv6 Policies.

4.1.4.1. Probe Query Message for SR-MPLS Policy

The probe query messages for end-to-end performance measurement of an SR-MPLS Policy is sent using its SR-MPLS header containing the MPLS segment list as shown in Figure 4.

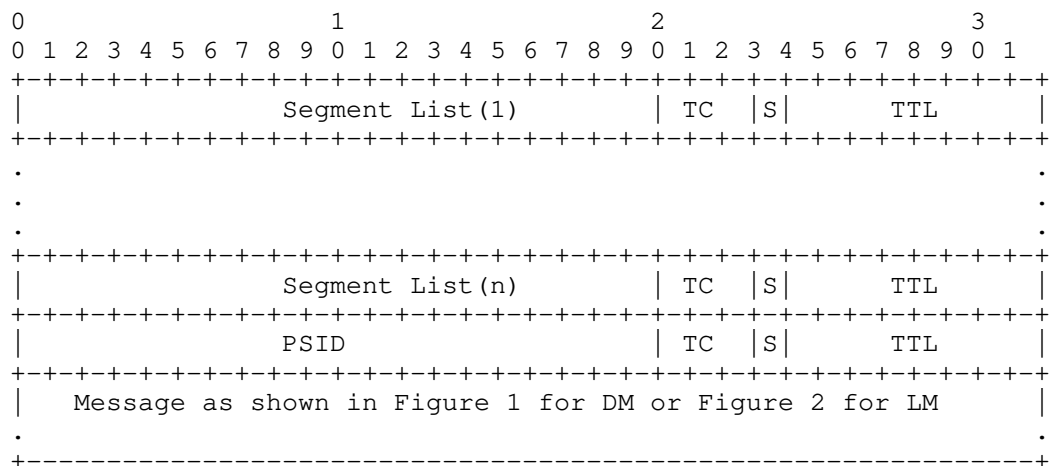


Figure 4: Probe Query Message for SR-MPLS Policy

The Segment List (SL) can be empty to indicate Implicit NULL label case for a single-hop SR Policy.

The Path Segment Identifier (PSID) [I-D.spring-mpls-path-segment] of the SR-MPLS Policy is used for accounting received traffic on the egress node for loss measurement. The PSID is not required for end-to-end SR Policy delay measurement.

4.1.4.2. Probe Query Message for SRv6 Policy

An SRv6 Policy setup using the SRv6 Segment Routing Header (SRH) and a Segment List as defined in [I-D.6man-segment-routing-header]. The probe query messages for end-to-end performance measurement of an SRv6 Policy is sent using its SRH and Segment List as shown in Figure 5.

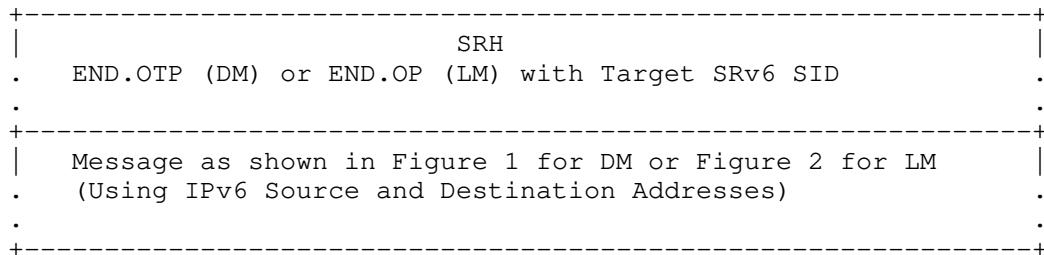
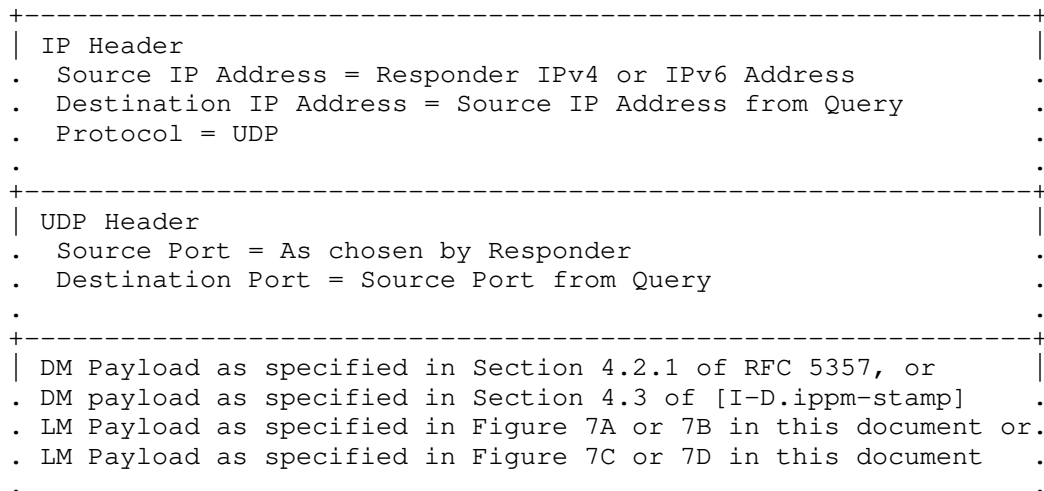


Figure 5: Probe Query Message for SRv6 Policy

For delay measurement of SRv6 Policy using SRH, END function END.OTP [I-D.6man-srv6-oam] is used with the target SRv6 SID to punt probe messages on the target node, as shown in Figure 5. Similarly, for loss measurement of SRv6 Policy, END function END.OP [I-D.6man-srv6-oam] is used with target SRv6 SID to punt probe messages on the target node.

4.2. Probe Response Message

The probe response message is sent using the IP/UDP information from the received probe query message. The content of the probe response message is shown in Figure 6.



+-----+

Figure 6: Probe Response Message

In this document, new probe response message formats are defined for loss measurement as shown in Figure 7A and Figure 7B. The message formats are hardware efficient due to the small size payload and well known locations of the counters. They are also similar to the delay measurement message formats.

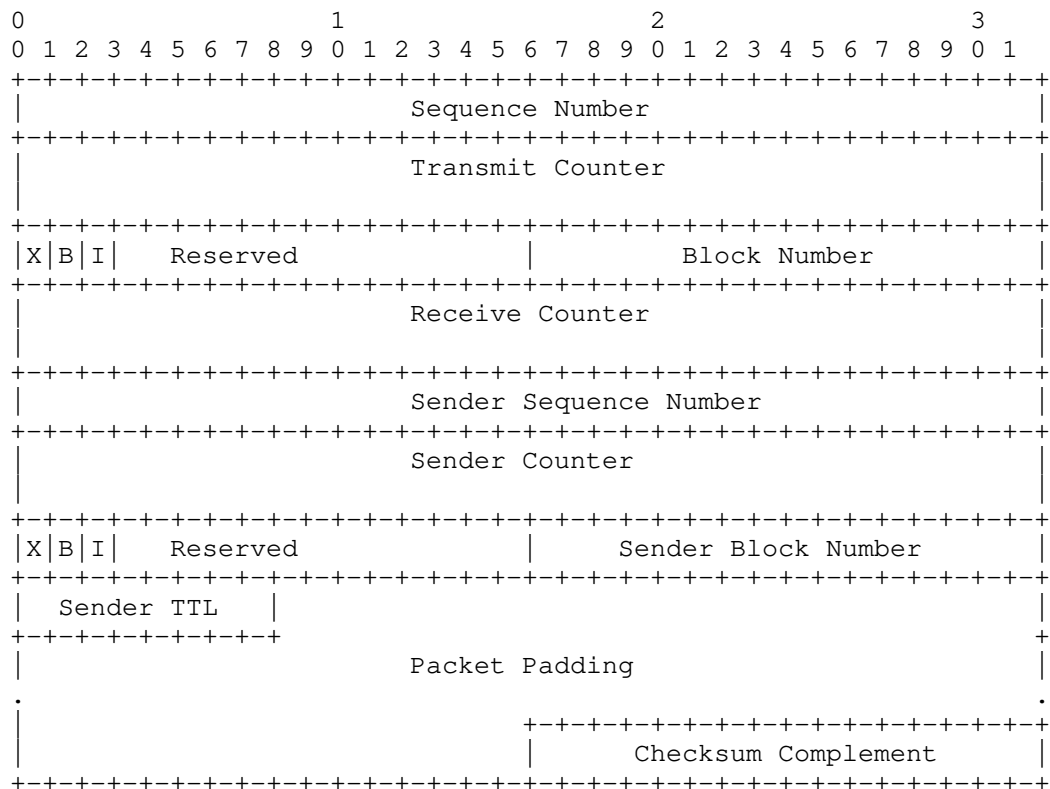
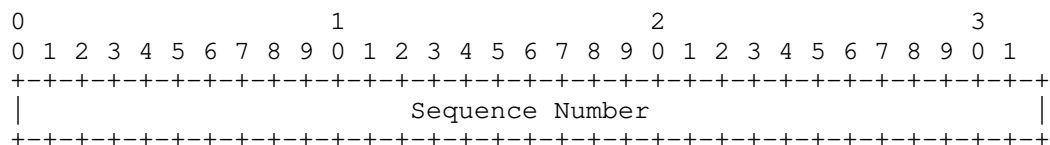


Figure 7A: LM Probe Response Message Format



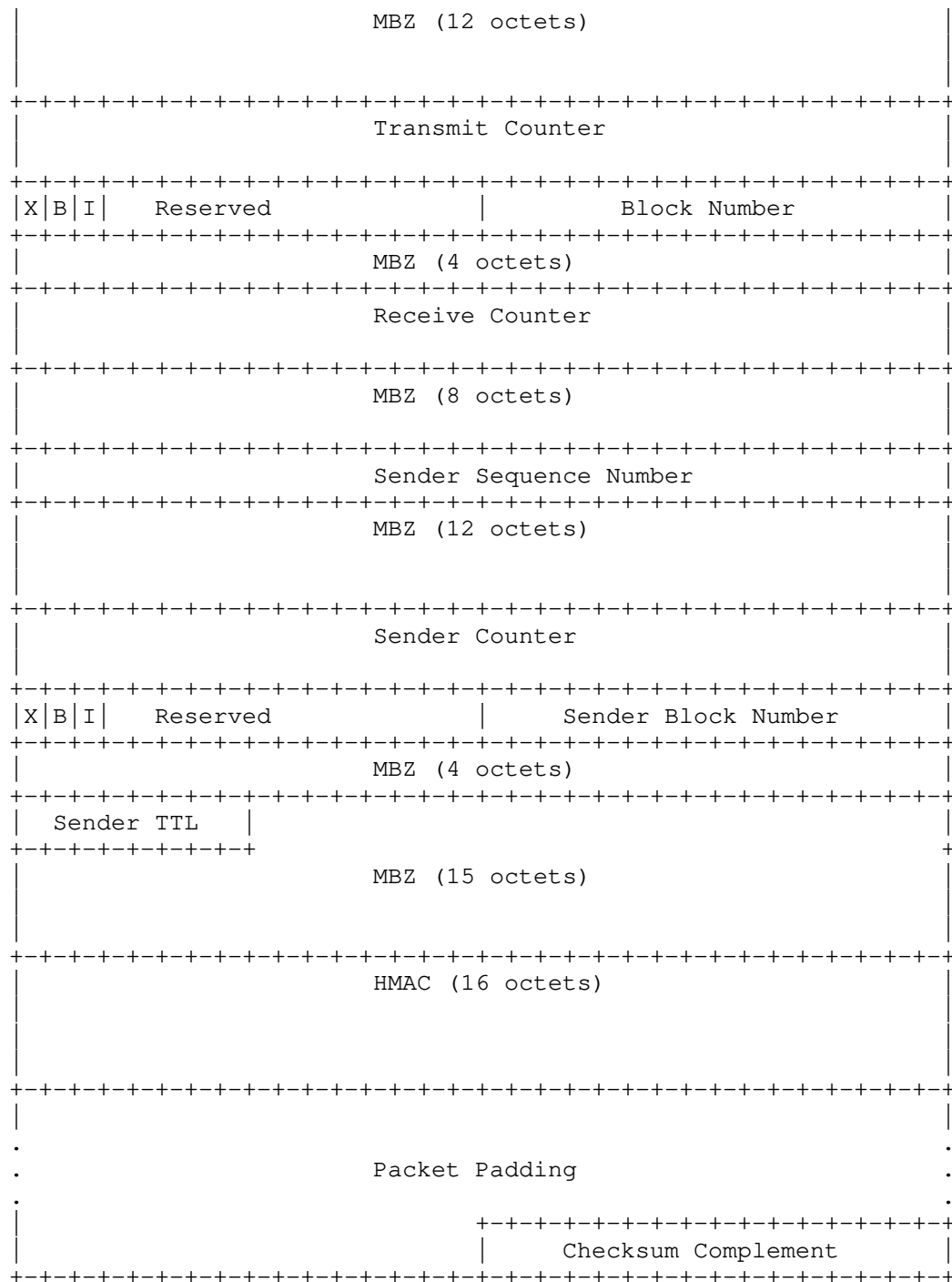


Figure 7B: LM Probe Response Message Format - Authenticated Mode

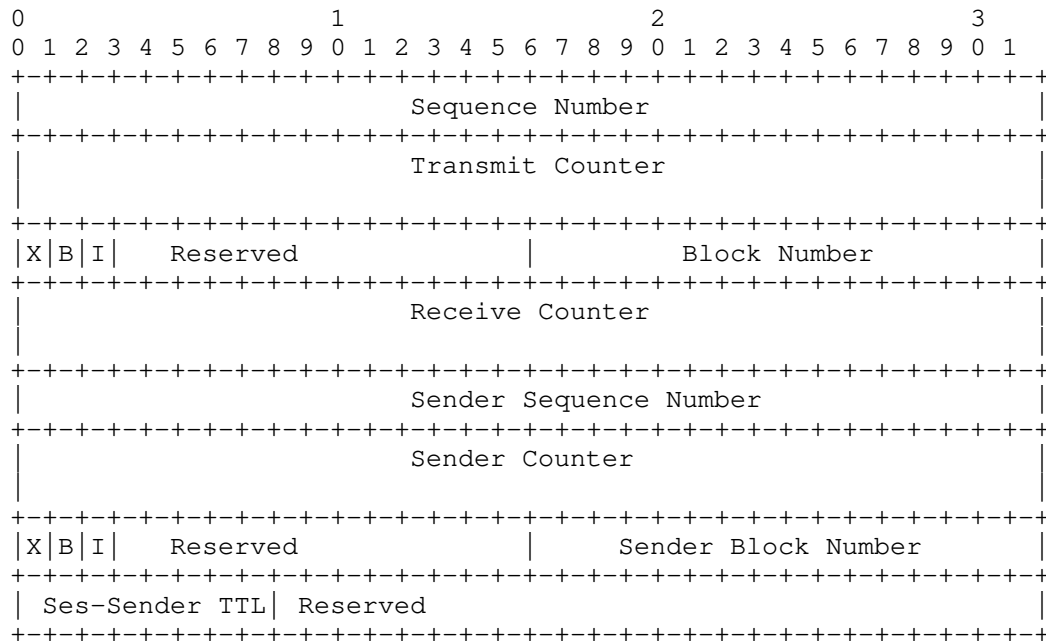
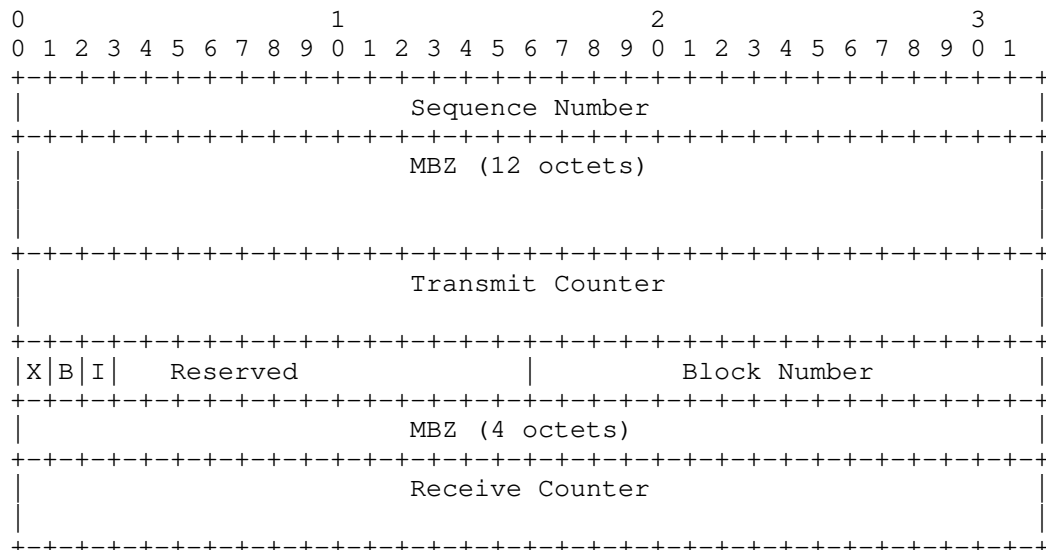


Figure 7C: STAMP LM Probe Response Message Format



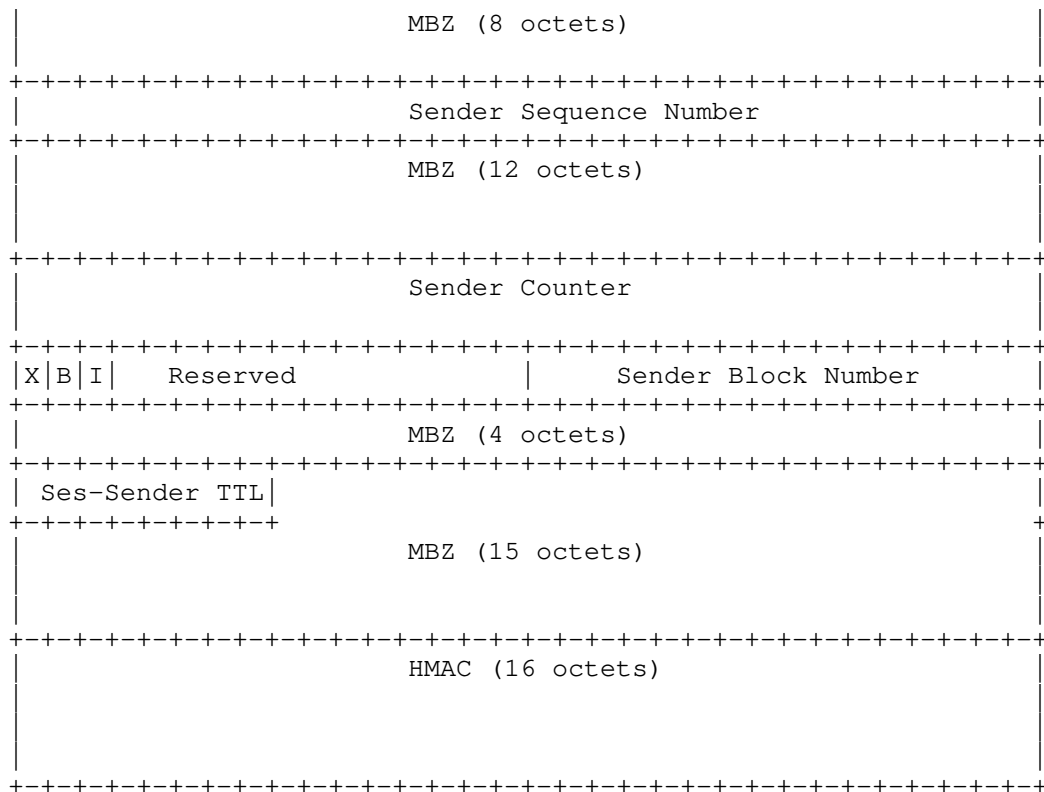


Figure 7D: STAMP LM Probe Response Message Format - Authenticated

4.2.1. One-way Measurement Mode

In one-way performance measurement mode, the probe response message as defined in Figure 6 is sent back out of band to the sender node, for both SR links and SR Policies.

4.2.2. Two-way Measurement Mode

In two-way performance measurement mode, when using a bidirectional path, the probe response message as defined in Figure 6 is sent back on the congruent path of the data traffic to the sender node, for both SR links and SR Policies.

4.2.2.1. Return Path TLV

For two-way performance measurement, the responder node needs to send the probe response message on a specific reverse path. This way the

destination node does not require any additional state. The sender node can request in the probe query message to the responder node to send a response back on a given reverse path (e.g. co-routed path for two-way measurement).

[I-D.ippm-stamp-option-tlv] defines STAMP probe query messages that can include one or more optional TLVs. New TLV Type (TBA1) is defined in this document for Return Path to carry reverse path for probe response messages (in the payload of the message). The format of the Return Path TLV is shown in Figure 8A and Figure 8B:

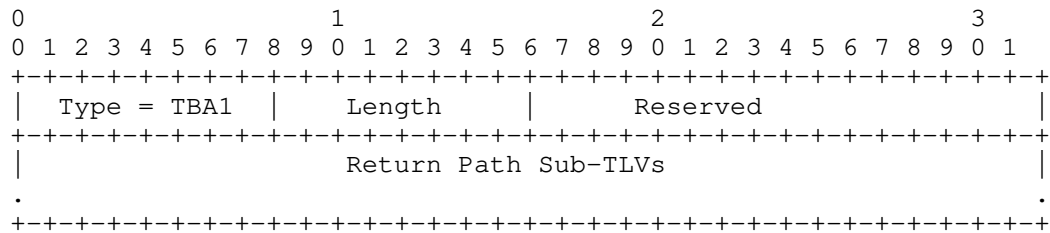


Figure 8A: Return Path TLV

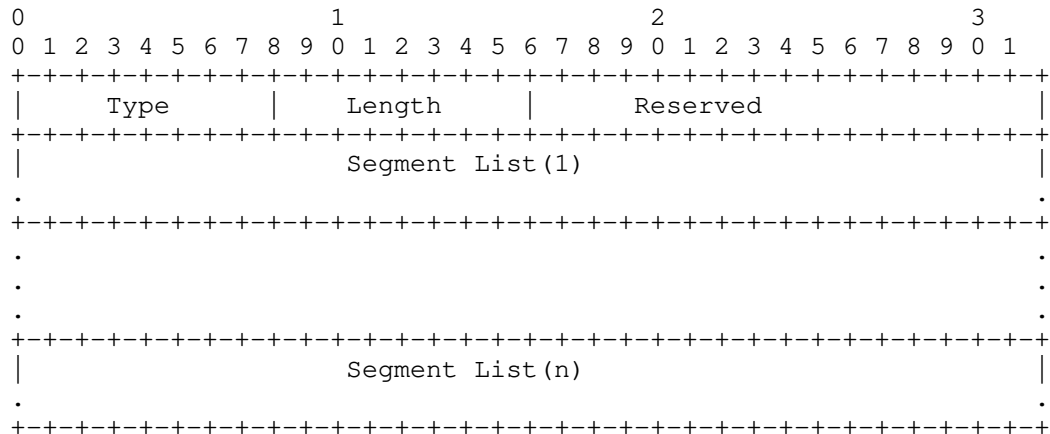


Figure 8B: Segment List Sub-TLV in Return Path TLV

The Segment List Sub-TLV in the Return Path TLV can be one of the following Types:

- o Type (value 1): Respond back on Incoming Interface (Layer-3 and Layer-2) (Segment List is Empty)

- o Type (value 2): SR-MPLS Segment List (Label Stack) of the Reverse SR Path
- o Type (value 3): SR-MPLS Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy
- o Type (value 4): SRv6 Segment List of the Reverse SR Path
- o Type (value 5): SRv6 Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy

The Return Path TLV is optional. The PM sender node MUST only insert one Return Path TLV in the probe query message and the responder node MUST only process the first Return Path TLV in the probe query message and ignore other Return Path TLVs if present. The responder node MUST send probe response message back on the reverse path specified in the Return Path TLV and MUST NOT add Return Path TLV in the probe response message.

4.2.2.2. Probe Response Message for SR-MPLS Policy

The message content for sending probe response message for two-way end-to-end performance measurement of an SR-MPLS Policy is shown in Figure 9.

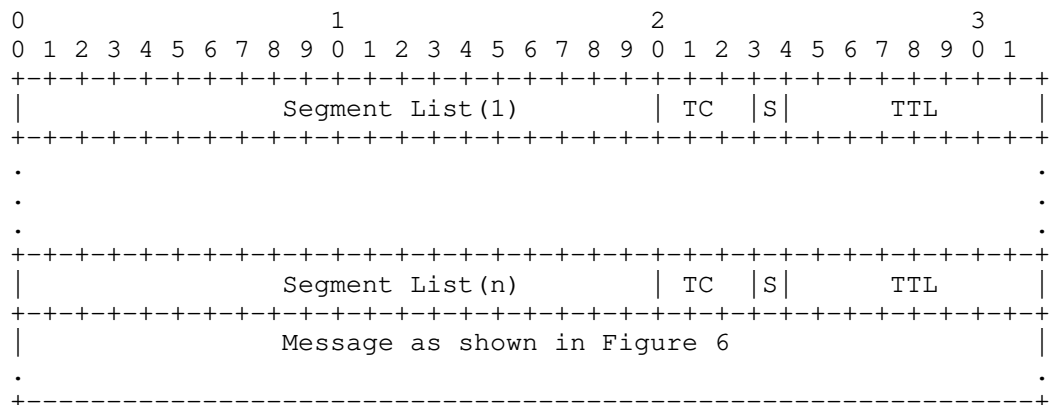


Figure 9: Probe Response Message for SR-MPLS Policy

The Path Segment Identifier (PSID) [I-D.spring-mpls-path-segment] of the forward SR Policy can be used to find the reverse SR Policy to send the probe response message for two-way measurement of SR Policy.

4.2.2.3. Probe Response Message for SRv6 Policy

The message content for sending probe response message on the congruent path of the data traffic for two-way end-to-end performance measurement of an SRv6 Policy with SRH is shown in Figure 10.

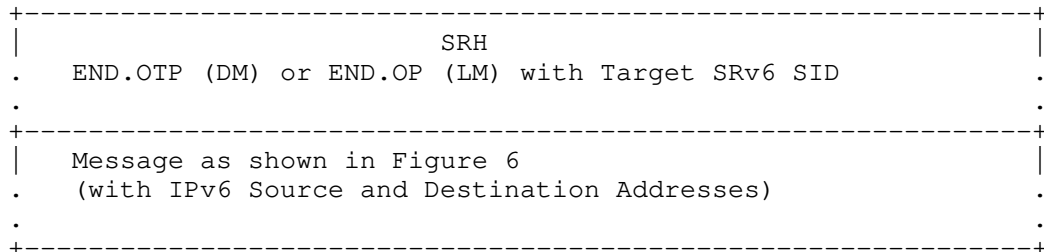


Figure 10: Probe Response Message for SRv6 Policy

4.2.3. Loopback Measurement Mode

The Loopback measurement mode can be used to measure round-trip delay for a bidirectional SR Path. The IP header of the probe query message contains the destination address equals to the sender address and the source address equals to the responder address. Optionally, the probe query message can carry the reverse path information (e.g. reverse path label stack for SR-MPLS) as part of the SR header. The responder node does not process the PM probe messages and generate response messages.

5. Performance Measurement for P2MP SR Policies

The procedures for delay and loss measurement described in this document for Point-to-Point (P2P) SR Policies [I-D.spring-segment-routing-policy] are also equally applicable to the Point-to-Multipoint (P2MP) SR Policies [I-D.spring-sr-p2mp-policy] as following:

- o The sender root node sends probe query messages using the Replication Segment defined in [I-D.spring-sr-p2mp-policy] for the P2MP SR Policy as shown in Figure 11.
- o Each responder leaf node sends its IP address in the Source Address of the probe response messages. This allows the sender root node to identify the responder leaf nodes of the P2MP SR Policy.
- o The P2MP root node measures the end-to-end delay and loss performance for each P2MP leaf node of the P2MP SR Policy.

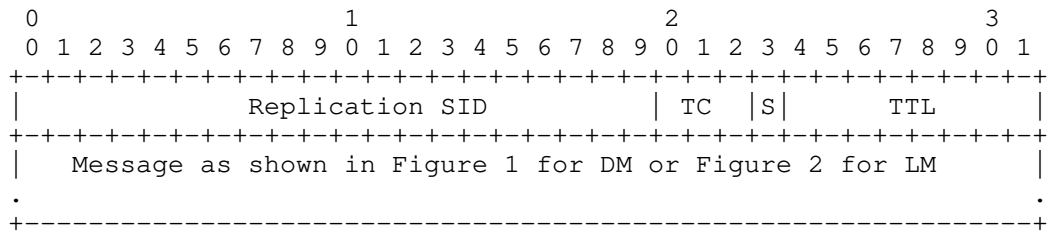


Figure 11: With Replication Segment for SR-MPLS Policy

6. ECMP Support for SR Policies

An SR Policy can have ECMPs between the source and transit nodes, between transit nodes and between transit and destination nodes. Usage of Anycast SID [RFC8402] by an SR Policy can result in ECMP paths via transit nodes part of that Anycast group. The PM probe messages need to be sent to traverse different ECMP paths to measure performance delay of an SR Policy.

Forwarding plane has various hashing functions available to forward packets on specific ECMP paths. The mechanisms described in [RFC8029] and [RFC5884] for handling ECMPs are also applicable to the performance measurement. In the IP header of the PM probe messages, Destination Addresses in 127/8 range for IPv4 or 0:0:0:0:0:FFFF:7F00/104 range for IPv6 can be used to exercise a particular ECMP path. As specified in [RFC6437], 3-tuple of Flow Label, Source Address and Destination Address fields in the IPv6 header can also be used.

7. Additional Message Processing Rules

7.1. TTL Value

The TTL or the Hop Limit field in the IP, MPLS and SRH headers of the probe query messages are set to 255 [RFC5357].

When using the Destination IPv4 Address from the 127/8 range, the TTL in the IPv4 header is set to 1 [RFC8029]. Similarly, when using the Destination IPv6 Address from the 0:0:0:0:0:FFFF:7F00/104 range, the Hop Limit field in the inner IPv6 header is set to 1 whereas in the outer IPv6 header is set to 255.

7.2. Router Alert Option

The Router Alert IP option is not set when using the routable

Destination IP Address in the probe messages.

When using the Destination IPv4 Address from the 127/8 range, to be able to punt probe packets on the responder node, the Router Alert IP Option of value 0x0 [RFC2113] for IPv4 MAY be added [RFC8029]. Similarly, when using the Destination IPv6 Address from the 0:0:0:0:0:FFFF:7F00/104 range, the Router Alert IP Option of value 69 [RFC7506] for IPv6 MAY be added in the destination option. For SRv6 Policy, it is added in the inner IPv6 header.

7.3. UDP Checksum

The Checksum Complement for delay and loss measurement messages follows the procedure defined in [RFC7820] and can be optionally used with the procedures defined in this document.

For IPv4 and IPv6 probe messages, where the hardware is not capable of re-computing the UDP checksum or adding checksum complement [RFC7820], the sender node sets the UDP checksum to 0 [RFC6936] [RFC8085]. The receiving node bypasses the checksum validation and accepts the packets with UDP checksum of 0 for the UDP port being used for PM.

8. Security Considerations

The performance measurement is intended for deployment in well-managed private and service provider networks. As such, it assumes that a node involved in a measurement operation has previously verified the integrity of the path and the identity of the far-end responder node.

If desired, attacks can be mitigated by performing basic validation and sanity checks, at the sender, of the counter or timestamp fields in received measurement response messages. The minimal state associated with these protocols also limits the extent of measurement disruption that can be caused by a corrupt or invalid message to a single query/response cycle.

Use of HMAC-SHA-256 in the authenticated mode protects the data integrity of the probe messages. SRv6 has HMAC protection authentication defined for SRH [I-D.6man-segment-routing-header]. Hence, PM probe messages for SRv6 may not need authentication mode. Cryptographic measures may be enhanced by the correct configuration of access-control lists and firewalls.

9. IANA Considerations

IANA is requested to allocate value for the following Return Path TLV Type for [I-D.ippm-stamp-option-tlv] to be carried in PM probe query messages:

- o Type TBA1: Return Path TLV

IANA is also requested to allocate the values for the following Sub-TLV Types for the Return Path TLV.

- o Type (value 1): Respond back on Incoming Interface (Layer-3 and Layer-2) (Segment List is Empty)
- o Type (value 2): SR-MPLS Segment List (Label Stack) of the Reverse SR Path
- o Type (value 3): SR-MPLS Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy
- o Type (value 4): SRv6 Segment List of the Reverse SR Path
- o Type (value 5): SRv6 Binding SID [I-D.pce-binding-label-sid] of the Reverse SR Policy

10. References

10.1. Normative References

- [RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.
- [I-D.6man-srv6-oam] Ali, Z., et al., "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ietf-6man-spring-srv6-oam,

work in progress.

[I-D.ippm-stamp] Mirsky, G. et al. "Simple Two-way Active Measurement Protocol", draft-ietf-ippm-stamp, work in progress.

[I-D.ippm-stamp-option-tlv] Mirsky, G., et al., "Simple Two-way Active Measurement Protocol Optional Extensions", draft-ietf-ippm-stamp-option-tlv, work in progress.

10.2. Informative References

- [IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010.
- [RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, October, 2010
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

- [RFC7506] Raza, K., Akiya, N., and C. Pignataro, "IPv6 Router Alert Option for MPLS Operations, Administration, and Maintenance (OAM)", RFC 7506, DOI 10.17487/RFC7506, April 2015, <<http://www.rfc-editor.org/info/rfc7506>>.
- [RFC7820] Mizrahi, T., "UDP Checksum Complement in the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7820, March 2016.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Kumar, N., Aldrin, S. and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, March 2017.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<http://www.rfc-editor.org/info/rfc8085>>.
- [RFC8186] Mirsky, G., and I. Meilik, "Support of the IEEE 1588 Timestamp Format in a Two-Way Active Measurement Protocol (TWAMP)", RFC 8186, June 2017.
- [RFC8321] Fioccola, G. Ed., "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, January 2018.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy, work in progress.
- [I-D.spring-sr-p2mp-policy] Voyer, D. Ed., et al., "SR Replication Segment for Multi-point Service Delivery", draft-voyer-spring-sr-replication-segment, work in progress.
- [I-D.spring-mpls-path-segment] Cheng, W., et al., "Path Segment in MPLS Based Segment Routing Network", draft-ietf-spring-mpls-path-segment, work in progress.
- [I-D.6man-segment-routing-header] Filsfils, C., et al., "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header, work in progress.

- [I-D.pce-binding-label-sid] Filsfils, C., et al., "Carrying Binding Label/ Segment-ID in PCE-based Networks", draft-ietf-pce-binding-label-sid, work in progress.
- [BBF.TR-390] "Performance Measurement from IP Edge to Customer Equipment using TWAMP Light", BBF TR-390, May 2017.
- [I-D.spring-ioam-sr-mpls] Gandhi, R. Ed., et al., "Segment Routing with MPLS Data Plane Encapsulation for In-situ OAM Data", draft-gandhi-spring-ioam-sr-mpls, work in progress.
- [I-D.spring-ioam-srv6] Ali, Z., et al., "Segment Routing Header encapsulation for In-situ OAM Data", draft-ali-spring-ioam-srv6, work in progress.

Acknowledgments

The authors would like to thank Thierry Couture for the discussions on the use-cases for TWAMP Light in Segment Routing. The authors would also like to thank Greg Mirsky for reviewing this document and providing useful comments and suggestions. Patrick Khordoc and Radu Valceanu, both from Cisco Systems have helped significantly improve the mechanisms defined in this document. The authors would like to acknowledge the earlier work on the loss measurement using TWAMP described in draft-xiao-ippm-twamp-ext-direct-loss.

Authors' Addresses

Rakesh Gandhi (editor)
Cisco Systems, Inc.
Canada
Email: rgandhi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Email: cfilsfil@cisco.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Bart Janssens

Colt

Email: Bart.Janssens@colt.net

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2021

R. Gandhi, Ed.
C. Filsfils
Cisco Systems, Inc.
D. Voyer
Bell Canada
M. Chen
Huawei
B. Janssens
Colt
October 21, 2020

Performance Measurement Using TWAMP Light for Segment Routing Networks
draft-gandhi-spring-twamp-srpm-11

Abstract

Segment Routing (SR) leverages the source routing paradigm. SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. This document specifies procedure for sending and processing probe query and response messages for Performance Measurement (PM) in Segment Routing networks. The procedure uses the mechanisms defined in RFC 5357 (Two-Way Active Measurement Protocol (TWAMP) Light) and its extensions for Performance Measurement. The procedure specified is applicable to SR-MPLS and SRv6 data planes and is used for both Links and end-to-end SR Paths including SR Policies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Conventions Used in This Document | 3 |
| 2.1. Requirements Language | 3 |
| 2.2. Abbreviations | 3 |
| 2.3. Reference Topology | 4 |
| 3. Overview | 5 |
| 3.1. Example Provisioning Model | 6 |
| 4. Probe Messages | 7 |
| 4.1. Probe Query Message | 7 |
| 4.1.1. Delay Measurement Query Message | 7 |
| 4.1.2. Loss Measurement Query Message | 8 |
| 4.1.3. Probe Query for Links | 9 |
| 4.1.4. Probe Query for SR Policy | 9 |
| 4.2. Probe Response Message | 11 |
| 4.2.1. One-way Measurement Mode | 11 |
| 4.2.2. Two-way Measurement Mode | 11 |
| 4.2.3. Loopback Measurement Mode | 13 |
| 4.3. Additional Probe Message Processing Rules | 14 |
| 4.3.1. TTL and Hop Limit | 14 |
| 4.3.2. Router Alert Option | 14 |
| 4.3.3. UDP Checksum | 14 |
| 5. Performance Measurement for P2MP SR Policies | 14 |
| 6. ECMP Support for SR Policies | 16 |
| 7. Performance Delay and Liveness Monitoring | 16 |
| 8. Security Considerations | 16 |
| 9. IANA Considerations | 17 |
| 10. References | 17 |
| 10.1. Normative References | 17 |
| 10.2. Informative References | 17 |
| Acknowledgments | 20 |
| Authors' Addresses | 21 |

1. Introduction

Segment Routing (SR) leverages the source routing paradigm and greatly simplifies network operations for Software Defined Networks (SDNs). SR is applicable to both Multiprotocol Label Switching (SR-MPLS) and IPv6 (SRv6) data planes. SR takes advantage of the Equal-Cost Multipaths (ECMPs) between source and transit nodes, between transit nodes and between transit and destination nodes. SR Policies as defined in [I-D.ietf-spring-segment-routing-policy] are used to steer traffic through a specific, user-defined paths using a stack of Segments. Built-in SR Performance Measurement (PM) is one of the essential requirements to provide Service Level Agreements (SLAs).

The One-Way Active Measurement Protocol (OWAMP) defined in [RFC4656] and Two-Way Active Measurement Protocol (TWAMP) defined in [RFC5357] provide capabilities for the measurement of various performance metrics in IP networks using probe messages. These protocols rely on control-channel signaling to establish a test-channel over an UDP path. The TWAMP Light [Appendix I in RFC5357] [BBF.TR-390] provides simplified mechanisms for active performance measurement in Customer IP networks by provisioning UDP paths and eliminates the need for control-channel signaling.

This document specifies procedures for sending and processing probe query and response messages for Performance Measurement in SR networks. The procedure uses the mechanisms defined in [RFC5357] (TWAMP Light) and its extensions for Performance Measurement. The procedure specified is applicable to SR-MPLS and SRv6 data planes and is used for both Links and end-to-end SR Paths including SR Policies and Flex- Algo IGP Paths. Unless otherwise specified, the mechanisms defined in [RFC5357] are not modified by this document.

2. Conventions Used in This Document

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

BSID: Binding Segment ID.

DM: Delay Measurement.

ECMP: Equal Cost Multi-Path.

HMAC: Hashed Message Authentication Code.

LM: Loss Measurement.

MPLS: Multiprotocol Label Switching.

NTP: Network Time Protocol.

OWAMP: One-Way Active Measurement Protocol.

PM: Performance Measurement.

PSID: Path Segment Identifier.

PTP: Precision Time Protocol.

SID: Segment ID.

SL: Segment List.

SR: Segment Routing.

SRH: Segment Routing Header.

SR-MPLS: Segment Routing with MPLS data plane.

SRv6: Segment Routing with IPv6 data plane.

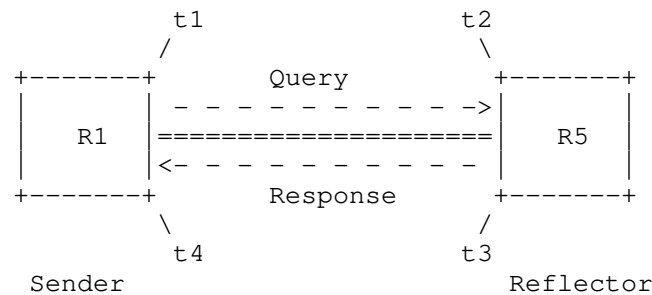
TC: Traffic Class.

TWAMP: Two-Way Active Measurement Protocol.

2.3. Reference Topology

In the reference topology shown below, the sender node R1 initiates a performance measurement probe query message and the reflector node R5 sends a probe response message for the query message received. The probe response message is typically sent to the sender node R1.

SR is enabled on nodes R1 and R5. The nodes R1 and R5 may be directly connected via a Link or there exists a Point-to-Point (P2P) SR Path e.g. SR Policy [I-D.ietf-spring-segment-routing-policy] on node R1 (called head-end) with destination to node R5 (called tail-end).



Reference Topology

3. Overview

For one-way and two-way delay measurements in Segment Routing networks, the probe messages defined in [RFC5357] are used. For direct-mode and inferred-mode loss measurements, the probe messages defined in [I-D.gandhi-ippm-twamp-srpm] are used. For both Links and end-to-end SR Paths including SR Policies and Flex-Algo IGP Paths, no PM state for delay or loss measurement need to be created on the reflector node R5.

Separate UDP destination port numbers are user-configured for delay and loss measurements. As specified in [RFC8545], the reflector supports the destination UDP port 862 for delay measurement probe messages by default. This UDP port however, is not used for loss measurement probe messages. The sender uses the UDP port number following the guidelines specified in Section 6 in [RFC6335]. The same destination UDP port is used for Links and SR Paths and the reflector is unaware if the query is for the Links or SR Paths. The number of UDP ports with PM functionality needs to be minimized due to limited hardware resources.

For Performance Measurement, probe query and response messages are sent as following:

- o For delay measurement, the probe messages are sent on the congruent path of the data traffic by the sender node, and are used to measure the delay experienced by the actual data traffic flowing on the Links and SR Paths.
- o For loss measurement, the probe messages are sent on the congruent path of the data traffic by the sender node, and are used to collect the receive traffic counters for the incoming link or incoming SID where the probe query messages are received at the reflector node (incoming link or incoming SID needed since the reflector node does not have PM state present).

The In-Situ Operations, Administration, and Maintenance (IOAM) mechanisms for SR-MPLS defined in [I-D.gandhi-mpls-ioam-sr] and for SRv6 defined in [I-D.ali-spring-ioam-srv6] are used to carry PM information such as timestamp in-band as part of the data packets, and are outside the scope of this document.

3.1. Example Provisioning Model

An example of a provisioning model and typical measurement parameters for each user-configured destination UDP port for performance delay and loss measurements is shown in the following Figure 1:

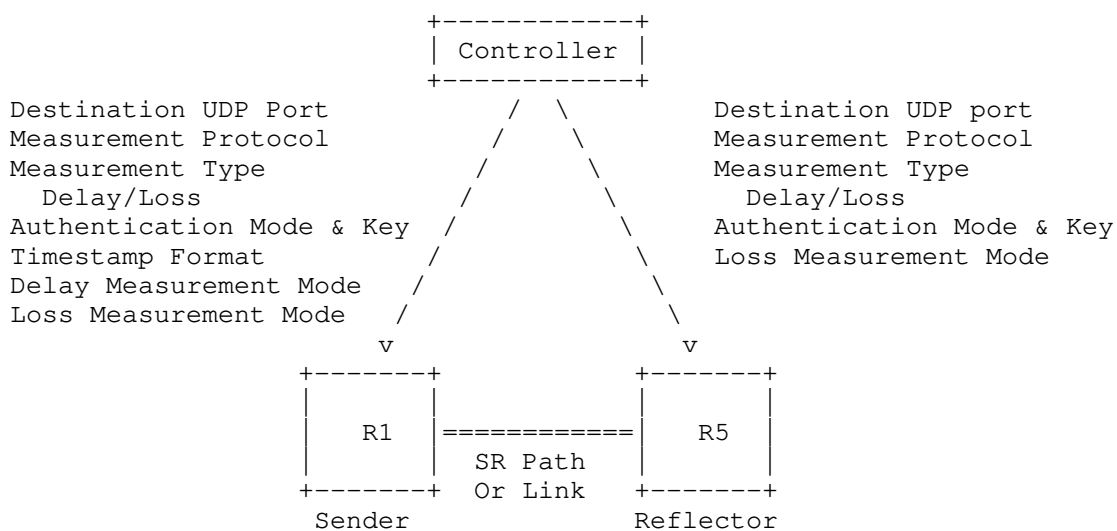


Figure 1: Example Provisioning Model

Example of Measurement Protocol is TWAMP Light, example of the Timestamp Format is PTPv2 [IEEE1588] or NTP and example of the Loss Measurement mode is inferred-mode or direct-mode.

The mechanisms to provision the sender and reflector nodes are outside the scope of this document. The provisioning model is not used for signaling the PM parameters between the reflector and sender nodes in SR networks.

The reflector node R5 uses the parameters for the timestamp format and delay measurement mode (i.e. one-way or two-way mode) from the received probe query message.

4. Probe Messages

4.1. Probe Query Message

The probe messages defined in [RFC5357] are used for delay measurement for Links and end-to-end SR Paths including SR Policies. For loss measurement, the probe messages defined in [I-D.gandhi-ippm-twamp-srpm] are used.

4.1.1. Delay Measurement Query Message

The message content for delay measurement probe query message using UDP header [RFC0768] is shown in Figure 2. The DM probe query message is sent with user-configured Destination UDP port number for DM. The Destination UDP port cannot be used as Source port, since the message does not have any indication to distinguish between the query and response message. The payload of the DM probe query message contains the delay measurement message defined in Section 4.1.2 of [RFC5357]. For symmetrical size query and response messages as defined in [RFC6038], the DM probe query message contains the payload format defined in Section 4.2.1 of [RFC5357].

```

+-----+
| IP Header                                     |
. Source IP Address = Sender IPv4 or IPv6 Address .
. Destination IP Address = Reflector IPv4 or IPv6 Address .
. Protocol = UDP .
. .
+-----+
| UDP Header                                     |
. Source Port = As chosen by Sender .
. Destination Port = User-configured Port for Delay Measurement.
. .
+-----+
| Payload = DM Message as specified in Section 4.2.1 of RFC 5357 |
. Payload = DM Message as specified in Section 4.1.2 of RFC 5357.
. .
+-----+

```

Figure 2: DM Probe Query Message

Timestamp field is eight bytes and use the format defined in Section 4.2.1 of [RFC5357]. It is recommended to use the IEEE 1588v2 Precision Time Protocol (PTP) truncated 64-bit timestamp format [IEEE1588] as specified in [RFC8186], with hardware support in Segment Routing networks.

4.1.1.1. Delay Measurement Authentication Mode

When using the authenticated mode for delay measurement, the matching authentication type (e.g. HMAC-SHA-256) and key are user-configured on both the sender and reflector nodes. A separate user-configured destination UDP port is used for the delay measurement in authentication mode due to the different probe message format.

4.1.2. Loss Measurement Query Message

The message content for loss measurement probe query message using UDP header [RFC0768] is shown in Figure 3. The LM probe query message is sent with user-configured Destination UDP port number for LM, which is a different Destination UDP port number than DM. Separate Destination UDP ports are used for direct-mode and inferred-mode loss measurements. The Destination UDP port cannot be used as Source port, since the message does not have any indication to distinguish between the query and response message. The LM probe query message contains the payload for loss measurement as defined in [I-D.gandhi-ippm-twamp-srpm].

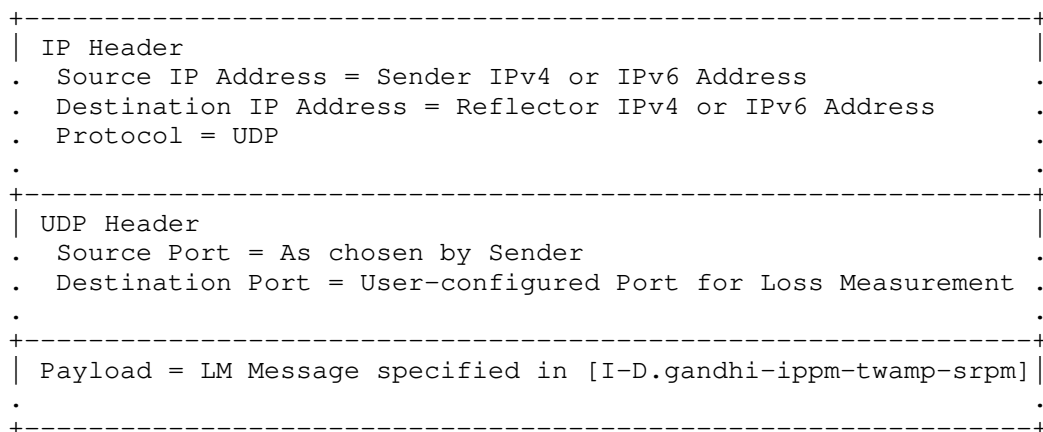


Figure 3: LM Probe Query Message

4.1.2.1. Loss Measurement Authentication Mode

When using the authenticated mode for loss measurement, the matching authentication type (e.g. HMAC-SHA-256) and key are user-configured on both the sender and reflector nodes. A separate user-configured destination UDP port is used for the loss measurement in authentication mode due to the different message format.

4.1.3. Probe Query for Links

The probe query message as defined in Figure 2 for delay measurement and Figure 3 for loss measurement are used for Links which may be physical, virtual or LAG (bundle), LAG (bundle) member, numbered/unnumbered Links. The probe messages are pre-routed over the Link for both delay and loss measurement. The local and remote IP addresses of the link are used as Source and Destination Addresses. They can also be IPv6 link local address as probe messages are pre-routed.

4.1.4. Probe Query for SR Policy

The performance delay and loss measurement for segment routing is applicable to both end-to-end SR-MPLS and SRv6 Policies.

The sender IPv4 or IPv6 address is used as the source address. The endpoint IPv4 or IPv6 address is used as the destination address. In the case of SR Policy with IPv4 endpoint of 0.0.0.0 or IPv6 endpoint of ::0 [I-D.ietf-spring-segment-routing-policy], the loopback address from range 127/8 for IPv4, or the loopback address ::1/128 for IPv6 is used as the destination address, respectively.

4.1.4.1. Probe Query Message for SR-MPLS Policy

The probe query messages for performance measurement of an end-to-end SR-MPLS Policy is sent using its SR-MPLS header containing the MPLS segment list as shown in Figure 4.

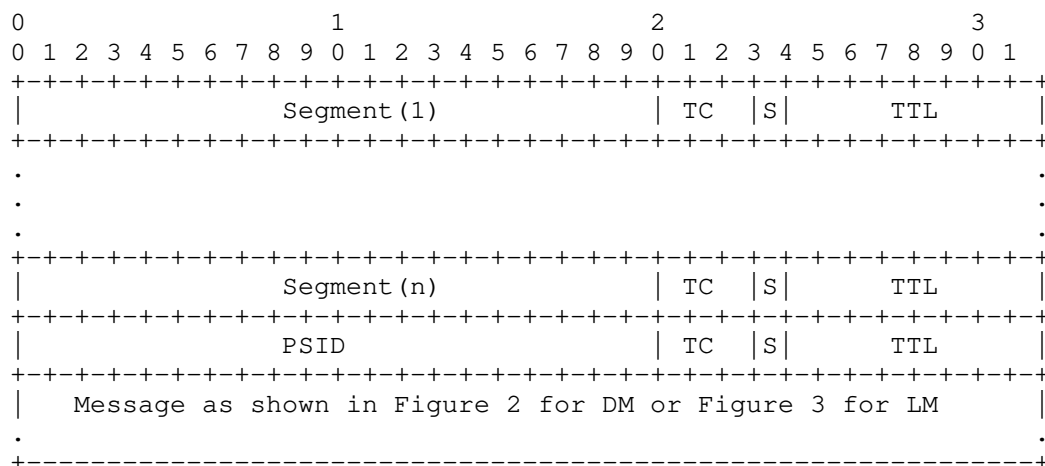


Figure 4: Example Probe Query Message for SR-MPLS Policy

The Segment List (SL) can be empty to indicate Implicit NULL label case for a single-hop SR Policy.

The Path Segment Identifier (PSID) [I-D.ietf-spring-mpls-path-segment] of the SR-MPLS Policy is used for accounting received traffic on the egress node for loss measurement.

4.1.4.2. Probe Query Message for SRv6 Policy

An SRv6 Policy setup using the SRv6 Segment Routing Header (SRH) and a Segment List as defined in [RFC8754]. The SRv6 network programming is defined in [I-D.ietf-spring-srv6-network-programming]. The probe query messages for performance measurement of an end-to-end SRv6 Policy is sent using its SRH with Segment List as shown in Figure 5. The procedure defined for upper-layer header processing for SRv6 SIDs in [I-D.ietf-spring-srv6-network-programming] is used to process the UDP header in the received probe query messages.

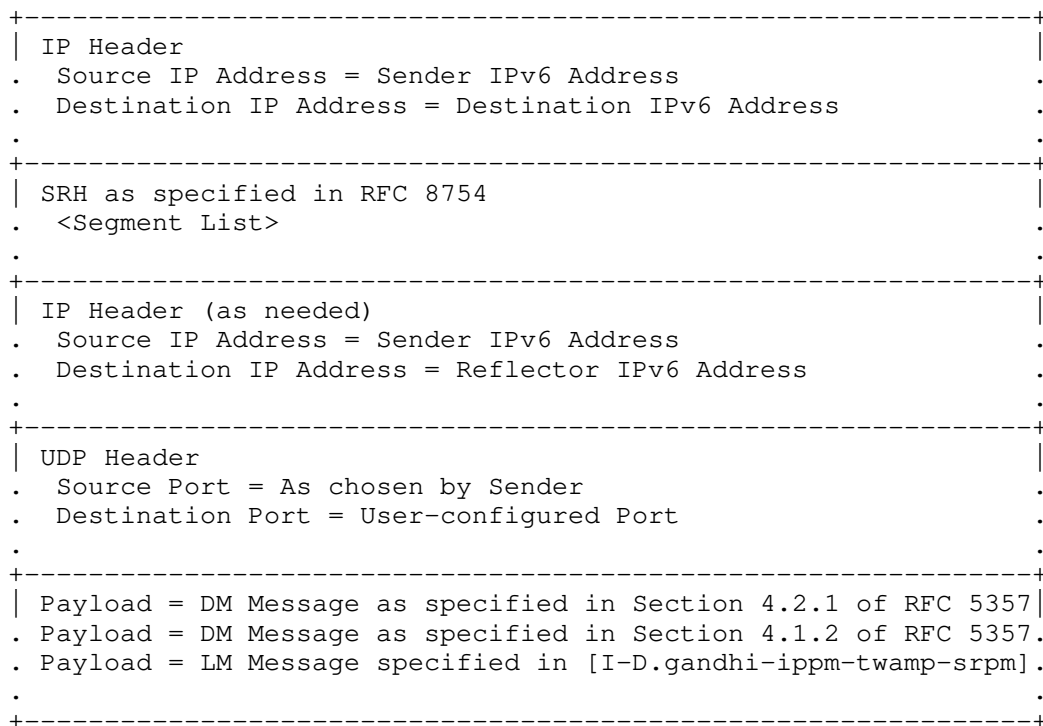


Figure 5: Example Probe Query Message for SRv6 Policy

4.2. Probe Response Message

The probe response message is sent using the IP/UDP information from the received probe query message. The content of the probe response message is shown in Figure 6.

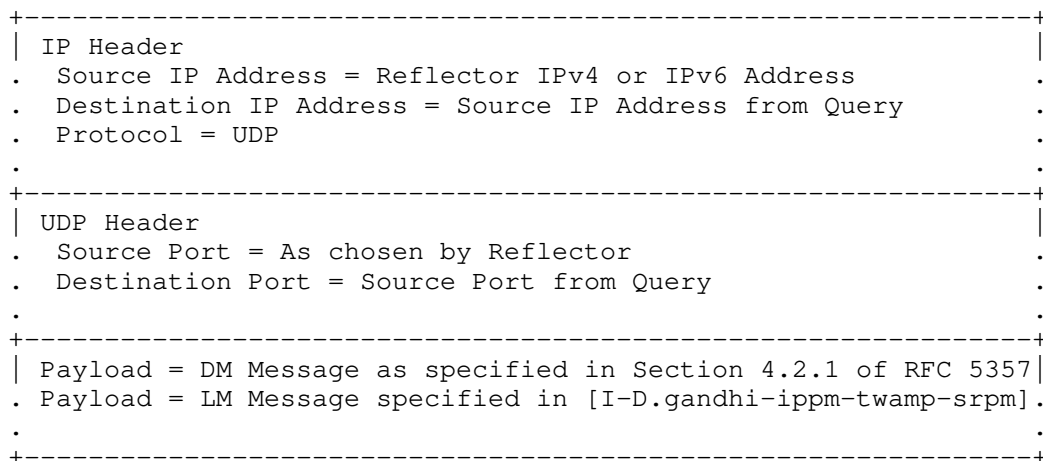


Figure 6: Probe Response Message

4.2.1. One-way Measurement Mode

In one-way measurement mode, the probe response message as defined in Figure 6 is sent back out-of-band to the sender node, for both Links and SR Policies. The Sender Control Code is set to "Out-of-band Response Requested". In this delay measurement mode, as per Reference Topology, all timestamps t_1 , t_2 , t_3 , and t_4 are collected by the probes. However, only timestamps t_1 and t_2 are used to measure one-way delay as $(t_2 - t_1)$.

4.2.2. Two-way Measurement Mode

In two-way measurement mode, when using a bidirectional path, the probe response message as defined in Figure 6 is sent back to the sender node on the congruent path of the data traffic on the same reverse direction Link or associated reverse SR Policy [I-D.ietf-pce-sr-bidir-path]. The Sender Control Code is set to "In-band Response Requested". In this delay measurement mode, as per Reference Topology, all timestamps t_1 , t_2 , t_3 , and t_4 are collected by the probes. All four timestamps are used to measure two-way delay as $((t_4 - t_1) - (t_3 - t_2))$.

4.2.2.1. Probe Response Message for SR-MPLS Policy

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------------------|---|---|---|---|---|---|---|---|---|----|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| Segment (1) | | | | | | | | | | TC | | S | | TTL | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Segment (n) | | | | | | | | | | TC | | S | | TTL | | | | | | | | | | | | | | | | | |
| Message as shown in Figure 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

The Path Segment Identifier (PSID) [I-D.ietf-spring-mpls-path-segment] of the forward SR Policy in the probe query can be used to find the associated reverse SR Policy [I-D.ietf-pce-sr-bidir-path] to send the probe response message for two-way measurement of SR Policy.

The message content for sending probe response message on the congruent path of the data traffic for two-way performance measurement of an end-to-end SRv6 Policy with SRH is shown in Figure 8. The procedure defined for upper-layer header processing for SRv6 SIDs in [I-D.ietf-spring-srv6-network-programming] is used to process the UDP header in the received probe response messages.


```

+-----+
| IP Header                                     |
. Source IP Address = Reflector IPv6 Address   .
. Destination IP Address = Destination IPv6 Address .
.                                             .
+-----+
| SRH as specified in RFC 8754                 |
. <Segment List>                             .
.                                             .
+-----+
| IP Header (as needed)                       |
. Source IP Address = Reflector IPv6 Address   .
. Destination IP Address = Source IPv6 Address from Query .
.                                             .
+-----+
| UDP Header                                   |
. Source Port = As chosen by Sender           .
. Destination Port = User-configured Port     .
.                                             .
+-----+
| Payload = DM Message as specified in Section 4.2.1 of RFC 5357 |
. Payload = LM Message specified in [I-D.gandhi-ippm-twamp-srpm].
.                                             .
+-----+

```

Figure 8: Example Probe Response Message for SRv6 Policy

4.2.3. Loopback Measurement Mode

The Loopback measurement mode can be used to measure round-trip delay for a bidirectional SR Path. The IP header of the probe query message contains the destination address equals to the sender address and the source address equals to the reflector address. Optionally, the probe query message can carry the reverse path information (e.g. reverse path label stack for SR-MPLS) as part of the SR header. The probe messages are not punted at the reflector node and it does not process them and generate response messages. The Sender Control Code is set to the default value of 0. In this mode, as the probe packet is not punted on the reflector node for processing, the querier copies the 'Sequence Number' in 'Session-Sender Sequence Number' directly. In this delay measurement mode, as per Reference Topology, the timestamps t1 and t4 are collected by the probes. Both these timestamps are used to measure round-trip delay as $(t4 - t1)$.

4.3. Additional Probe Message Processing Rules

The processing rules defined in this section are applicable to TWAMP Light messages for delay and loss measurement for Links and end-to-end SR Paths including SR Policies.

4.3.1. TTL and Hop Limit

The TTL field in the IPv4 and MPLS headers of the probe query messages is set to 255 [RFC5357]. Similarly, the Hop Limit field in the IPv6 and SRH headers of the probe query messages is set to 255 [RFC5357].

When using the Destination IPv4 Address from range 127/8, the TTL field in the IPv4 header is set to 1 [RFC8029]. Similarly, when using the Destination IPv6 Address from the ::FFFF:127/104 range, the Hop Limit field in the IPv6 header is set to 1.

For Link performance delay and loss measurements, the TTL or Hop Limit field in the probe message is set to 1 in both one-way and two-way measurement modes.

4.3.2. Router Alert Option

The Router Alert IP option (RAO) [RFC2113] is not set in the probe messages.

4.3.3. UDP Checksum

The UDP Checksum Complement for delay and loss measurement messages follows the procedure defined in [RFC7820] and can be optionally used with the procedures defined in this document.

For IPv4 and IPv6 probe messages, where the hardware is not capable of re-computing the UDP checksum or adding checksum complement [RFC7820], the sender node sets the UDP checksum to 0 [RFC6936] [RFC8085]. The receiving node bypasses the checksum validation and accepts the packets with UDP checksum value 0 for the UDP port being used for delay and loss measurements.

5. Performance Measurement for P2MP SR Policies

The Point-to-Multipoint (P2MP) SR Path that originates from a root node terminates on multiple destinations called leaf nodes (e.g. P2MP SR Policy [I-D.ietf-pim-sr-p2mp-policy] or P2MP Transport [I-D.shen-spring-p2mp-transport-chain]).

The procedures for delay and loss measurement described in this document for P2P SR Policies are also equally applicable to the P2MP SR Policies. The procedure for one-way measurement is defined as following:

- o The sender root node sends probe query messages using the Tree-SID defined in [I-D.ietf-pim-sr-p2mp-policy] for the P2MP SR-MPLS Policy as shown in Figure 9.
- o The probe query messages can contain the replication SID as defined in [I-D.ietf-spring-sr-replication-segment].
- o The Destination Address is set to the loopback address from range 127/8 for IPv4, or the loopback address ::1/128 for IPv6 address.
- o Each reflector leaf node sends its IP address in the Source Address of the probe response messages as shown in Figure 9. This allows the sender root node to identify the reflector leaf nodes of the P2MP SR Policy.
- o The P2MP root node measures the delay and loss performance for each P2MP leaf node of the end-to-end P2MP SR Policy.

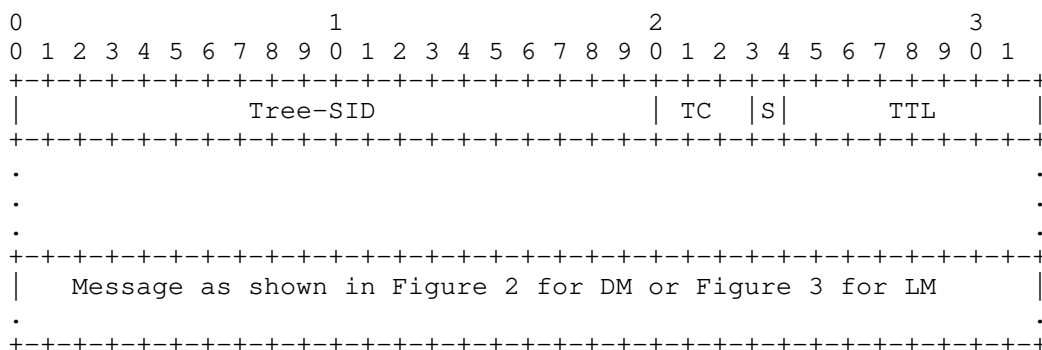


Figure 9: Example Probe Query with Tree-SID for SR-MPLS Policy

The probe query messages can also be sent using the scheme defined for P2MP Transport using Chain Replication that may contain Bud SID as defined in [I-D.shen-spring-p2mp-transport-chain].

The considerations for two-way mode for performance measurement for P2MP SR Policy (e.g. for bidirectional SR Path) are outside the scope of this document.

6. ECMP Support for SR Policies

An SR Policy can have ECMPs between the source and transit nodes, between transit nodes and between transit and destination nodes. Usage of Anycast SID [RFC8402] by an SR Policy can result in ECMP paths via transit nodes part of that Anycast group. The probe messages need to be sent to traverse different ECMP paths to measure performance delay of an SR Policy.

Forwarding plane has various hashing functions available to forward packets on specific ECMP paths. The mechanisms described in [RFC8029] and [RFC5884] for handling ECMPs are also applicable to the performance measurement. In IPv4 header of the probe messages, sweeping of Destination Address from range 127/8 can be used to exercise particular ECMP paths. As specified in [RFC6437], Flow Label field in the outer IPv6 header can also be used for sweeping.

The considerations for performance loss measurement for different ECMP paths of an SR Policy are outside the scope of this document.

7. Performance Delay and Liveness Monitoring

Liveness monitoring is required for connectivity verification and continuity check in an SR network. The procedure defined in this document for delay measurement using the TWAMP Light probe messages can also be applied to liveness monitoring of Links and SR Paths. The one-way or two-way measurement mode can be used for liveness monitoring. Liveness failure is notified when consecutive N number of probe response messages are not received back at the sender node, where N is locally provisioned value. Note that for one-way and two-way modes, the failure detection interval and scale for number of probe messages need to account for the processing of the probe query messages which need to be punted from the forwarding fast path (to slow path or control plane) and response messages need to be injected on the reflector node. This is improved by using the probes in loopback mode.

8. Security Considerations

The performance measurement is intended for deployment in well-managed private and service provider networks. As such, it assumes that a node involved in a measurement operation has previously verified the integrity of the path and the identity of the far-end reflector node.

If desired, attacks can be mitigated by performing basic validation and sanity checks, at the sender, of the counter or timestamp fields in received measurement response messages. The minimal state

associated with these protocols also limits the extent of measurement disruption that can be caused by a corrupt or invalid message to a single query/response cycle.

Use of HMAC-SHA-256 in the authenticated mode protects the data integrity of the probe messages. SRv6 has HMAC protection authentication defined for SRH [RFC8754]. Hence, probe messages for SRv6 may not need authentication mode. Cryptographic measures may be enhanced by the correct configuration of access-control lists and firewalls.

9. IANA Considerations

This document does not require any IANA action.

10. References

10.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [I-D.gandhi-ippm-twamp-srpm] Gandhi, R., Filsfils, C., Voyer, D., Chen, M., and B. Janssens, "TWAMP Light Extensions for Segment Routing", draft-gandhi-ippm-twamp-srpm-00 (work in progress), October 2020.

10.2. Informative References

- [IEEE1588] IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

- [RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, DOI 10.17487/RFC2113, February 1997, <<https://www.rfc-editor.org/info/rfc2113>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, DOI 10.17487/RFC6038, October 2010, <<https://www.rfc-editor.org/info/rfc6038>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC7820] Mizrahi, T., "UDP Checksum Complement in the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7820, DOI 10.17487/RFC7820, March 2016, <<https://www.rfc-editor.org/info/rfc7820>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.

- [RFC8186] Mirsky, G. and I. Meilik, "Support of the IEEE 1588 Timestamp Format in a Two-Way Active Measurement Protocol (TWAMP)", RFC 8186, DOI 10.17487/RFC8186, June 2017, <<https://www.rfc-editor.org/info/rfc8186>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8545] Morton, A., Ed. and G. Mirsky, Ed., "Well-Known Port Assignments for the One-Way Active Measurement Protocol (OWAMP) and the Two-Way Active Measurement Protocol (TWAMP)", RFC 8545, DOI 10.17487/RFC8545, March 2019, <<https://www.rfc-editor.org/info/rfc8545>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-08 (work in progress), July 2020.
- [I-D.ietf-spring-sr-replication-segment]
Voyer, D., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "SR Replication Segment for Multi-point Service Delivery", draft-ietf-spring-sr-replication-segment-00 (work in progress), July 2020.
- [I-D.shen-spring-p2mp-transport-chain]
Shen, Y., Zhang, Z., Parekh, R., Bidgoli, H., and Y. Kamite, "Point-to-Multipoint Transport Using Chain Replication in Segment Routing", draft-shen-spring-p2mp-transport-chain-02 (work in progress), April 2020.
- [I-D.ietf-pim-sr-p2mp-policy]
Voyer, D., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "Segment Routing Point-to-Multipoint Policy", draft-ietf-pim-sr-p2mp-policy-00 (work in progress), July 2020.

[I-D.ietf-spring-mpls-path-segment]

Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler,
"Path Segment in MPLS Based Segment Routing Network",
draft-ietf-spring-mpls-path-segment-03 (work in progress),
September 2020.

[I-D.ietf-spring-srv6-network-programming]

Filsfils, C., Camarillo, P., Leddy, J., Voyer, D.,
Matsushima, S., and Z. Li, "SRv6 Network Programming",
draft-ietf-spring-srv6-network-programming-24 (work in
progress), October 2020.

[BBF.TR-390]

"Performance Measurement from IP Edge to Customer
Equipment using TWAMP Light", BBF TR-390, May 2017.

[I-D.gandhi-mpls-ioam-sr]

Gandhi, R., Ali, Z., Filsfils, C., Brockners, F., Wen, B.,
and V. Kozak, "MPLS Data Plane Encapsulation for In-situ
OAM Data", draft-gandhi-mpls-ioam-sr-03 (work in
progress), September 2020.

[I-D.ali-spring-ioam-srv6]

Ali, Z., Gandhi, R., Filsfils, C., Brockners, F., Kumar,
N., Pignataro, C., Li, C., Chen, M., and G. Dawra,
"Segment Routing Header encapsulation for In-situ OAM
Data", draft-ali-spring-ioam-srv6-02 (work in progress),
November 2019.

[I-D.ietf-pce-sr-bidir-path]

Li, C., Chen, M., Cheng, W., Gandhi, R., and Q. Xiong,
"PCEP Extensions for Associated Bidirectional Segment
Routing (SR) Paths", draft-ietf-pce-sr-bidir-path-03 (work
in progress), September 2020.

Acknowledgments

The authors would like to thank Thierry Couture for the discussions on the use-cases for Performance Measurement in Segment Routing. The authors would also like to thank Greg Mirsky for reviewing this document and providing useful comments and suggestions. Patrick Khordoc and Radu Valceanu, both from Cisco Systems have helped significantly improve the mechanisms defined in this document.

Authors' Addresses

Rakesh Gandhi (editor)
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.

Email: cfilsfil@cisco.com

Daniel Voyer
Bell Canada

Email: daniel.voyer@bell.ca

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Bart Janssens
Colt

Email: Bart.Janssens@colt.net

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2020

X. Geng
M. Chen
Huawei
Y. Zhu
China Telecom
July 04, 2019

DetNet SRv6 Data Plane Encapsulation
draft-geng-detnet-dp-sol-srv6-01

Abstract

This document specifies Deterministic Networking data plane operation for SRv6 encapsulated user data.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Terminology and Conventions | 3 |
| 2.1. Terminology | 3 |
| 2.2. Conventions | 4 |
| 3. SRv6 DetNet Data Plane Overview | 4 |
| 3.1. SRv6 DetNet Data Plane Layers | 5 |
| 3.2. SRv6 DetNet Data Plane Scenarios | 5 |
| 4. SRv6 DetNet Data Plane Solution Considerations | 7 |
| 5. SRv6 DetNet Data Plane Solution for Service Sub-layer | 8 |
| 5.1. TLV Based SRv6 Data Plane Solution | 8 |
| 5.1.1. Encapsulation | 8 |
| 5.1.2. SRv6 Network Programming new Functions | 10 |
| 5.2. SID Based SRv6 Data Plane Solution | 11 |
| 5.2.1. Encapsulation | 11 |
| 5.2.2. Functions | 12 |
| 5.3. DetNet SID Based SRv6 Data Plane Solution | 13 |
| 5.3.1. Encapsulation | 13 |
| 5.3.2. Functions | 14 |
| 6. SRv6 DetNet Data Plane Solution for Transport Sub-layer | 14 |
| 7. IANA Considerations | 14 |
| 8. Security Considerations | 14 |
| 9. Acknowledgements | 14 |
| 10. Normative References | 14 |
| Authors' Addresses | 15 |

1. Introduction

Deterministic Networking (DetNet), as described in [I-D.ietf-detnet-architecture] provides a capability to carry specified data flows with extremely low data loss rates and bounded latency within a network domain. DetNet is enabled by a group of technologies, such as resource allocation, service protection and explicit routes.

Segment Routing(SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". SR can be applied over IPv6 data plane using the Segment Routing Extension Header (SRH, [I-D.ietf-6man-segment-routing-header]). A segment in segment routing terminology is not limited to a routing/forwarding function.

A segment can be associated to an arbitrary processing of the packet in the node identified by the segment. In other words, an SRv6 Segment can indicate functions that are executed locally in the node where they are defined. SRv6 network Programming [I-D.filsfils-spring-srv6-network-programming] describe the different segments and functions associated to them.

This document describes how to implement DetNet in an SRv6 enabled domain, including :

- o Source routing, which steers the DetNet flows through the network according to an explicit path with allocated resources;
- o Network programming, which applies instructions (functions) to packets in some special nodes (or even all the nodes) along the path in order to guarantee, e.g., service protection and congestion protection.

DetNet SRv6 encapsulation and new SRv6 functions ([I-D.filsfils-spring-srv6-network-programming]) for DetNet are defined in this document. Control plane and OAM are not in the scope of this document.

Control plane and OAM are not in the scope of this document.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1. Terminology

Terminologies for DetNet go along with the definition in [I-D.ietf-detnet-architecture] and [RFC8402]. Other terminologies are defined as follows:

- o NH: The IPv6 next-header field.
- o SID: A Segment Identifier ([RFC8402]).
- o SRH: The Segment Routing Header ([I-D.ietf-6man-segment-routing-header]).

2.2. Conventions

Conventions in the document are defined as follows:

- o NH=SRH means that NH is 43 with routing type 4 which is (as defined in [I-D.ietf-6man-segment-routing-header], the values representing the SRH.
- o A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.
- o SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3. It has to be noted that [I-D.ietf-6man-segment-routing-header] defines the segment list encoding in the reverse order of the path. A path represented by <S1,S2,S3>, will be encoded in the SRH as follows:

SegmentList[0]=S3

SegmentList[1]=S2

SegmentList[2]=S1

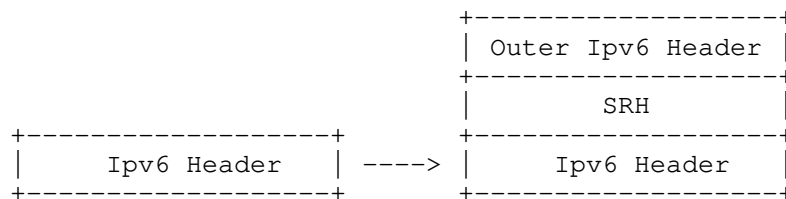
The reverse encoding has been defined in order to optimise the processing time of the segment list. See [draft-ietf-6man-segment-routing-header] for more details.

- o (SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:
 - IPv6 header with source and destination addresses SA and DA respectively, and next-header set to SRH (i.e.: 43 with type 4), with a list of segments(SIDs) <S1, S2, S3> with SegmentsLeft = SL
 - The payload of the packet is not represented
 - (S3, S2, S1; SL) represents the same SID list as <S1, S2, S3>, but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID

3. SRv6 DetNet Data Plane Overview

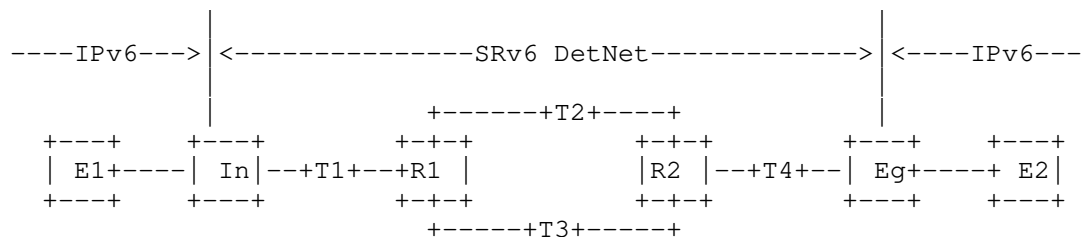
3.1. SRv6 DetNet Data Plane Layers

[I-D.ietf-detnet-architecture]decomposes the DetNet data plane into two sub-layers: service sub-layer and transport sub-layer. Different from DetNet MPLS data plane solution, which uses DetNet Control Word(d-CW) and S-Label to support service sub-layer and uses T-Label to support transport sub-layer, no explicit sub-layer division exists in SRv6 data plane. A classical SRv6 DetNet data plane solution is showed in the picture below:



The outer IPv6 Header with the SRH is used for carrying DetNet flows. Traffic Engineering is instantiated in the segment list of SRH, and other functions and arguments for service protection (packet replication, elimination and ordering) and congestion control (packet queuing and forwarding) are also defined in the SRH.

3.2. SRv6 DetNet Data Plane Scenarios



The figure above shows that an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the outer IPv6 header+SRH of the packet is popped, and the packet is sent to the destination E2.

The figure above shows that an IPv6 flow is sent our from the end station: E1. The packet of the flow is encapsulated as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the upper IPv6 header with SRH of the packet is popped, and the packet is transmitted to the destination(E2).

The DetNet packet processing is as follows:

Ingress:

Inserts the SRv6 Policy that will steer the packet from Ingress to the destination

The methods and mechanisms used for defining, instantiating and applying the policy are outside of this document. An example of policies are described in [I-D.ietf-spring-segment-routing-policy]

Flow Identification and Sequence Number are carried in the SRH.

Relay Node 1 (Replication Node):

Replicates the payload and IPv6 Header with the SRH. This is a new function in the context of SRv6 Network Programming which will associate a given SID to a replication instruction in the node originating and advertising the SID. The replication instruction includes:

- * The removal of the existing IPv6+SRH header
- * The encapsulation into a new outer IPv6+SRH header. Each packet (the original and the duplicated) are encapsulated into respectively new outer IPv6+SRH headers.

Binding two different SRv6 Policies respectively to the original packet and the replicated packet, which can steer the packets from Relay Node 1 to Relay Node 2 through two tunnels.

Relay Node 2 (Elimination Node):

Eliminates the redundant packets.

Binds a new SRv6 Policy to the survival packet, which steers the packet from Relay Node 2 to Egress.

Egress:

Decapsulates the outer Ipv6 header.

Sends the inter packet to the End Station 2.

The DetNet packet encapsulation is illustrated here below. It has to be noted that, in the example below, the R2 address is a SRH SID

associated to a TBD function related to the packet replication the node R1 has to perform. The same (or reverse) apply to node R2 which is in charge of the discard of the duplicated packet. Here also a new function will have a new SID allocated to it and representing the delete of the duplication in R2.

End Station1 output packet: (E1,E2)

Ingress output packet: (In, T1) (R1,T1, SL=2) (E1,E2)

Transit Node1 output packet: (In, R1) (R1,T1,SL=1) (E1,E2)

Relay Node1 output packets : (R1,T2) (R2,T2,SL=2) (E1,E2),
(R1,T3) (R2,T3,SL=2) (E1,E2)

Transit Node2 output packet: (R1, R2) (R2,T2,SL=1) (E1,E2)

Transit Node3 output packet: (R1, R2) (R2,T3,SL=1) (E1,E2)

Relay Node2 output packet: (R2, T4) (Eg,T4,SL=2) (E1,E2)

Transit Node4 output packet: (R2, Eg) (Eg,T4,SL=1) (E1,E2)

Egress out : (E1,E2)

4. SRv6 DetNet Data Plane Solution Considerations

To carry DetNet over SRv6, the following elements are required:

1. A method of identifying the SRv6 payload type;
2. A suitable explicit path to deliver the DetNet flow ;
3. A method of indicating packet processing, such as PREOF (Packet Replication, Elimination and Ordering as defined in [I-D.ietf-detnet-architecture]);
4. A method of identifying the DetNet flow;
5. A method of carrying DetNet sequence number;
6. A method of carrying queuing and forwarding indication to do congestion protection;

In this design, DetNet flows are encapsulated in an outer IPv6+SRH header at the Ingress Node. The SR policy identified in the SRH steers the DetNet flow along a selected path. The explicit path followed by a DetNet flow, which protect it from temporary

interruptions caused by the convergence of routing, is encoded within the SID list of the SR policy. The network device inside the DetNet domain forwards the packet according to IPv6 Destination Address (DA), and the IPv6 DA is updated with the SID List according to SRv6 forwarding procedures defined in [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming]

With SRv6 network programming, the SID list can also give instruments representing a function to be called at the node in the DetNet domain. Therefore DetNet specific functions defined in [I-D.ietf-detnet-architecture], corresponding to local packet processing in the network, can also be implemented by SRv6. New functions associated with SIDs for DetNet are defined in this document.

This document describes how DetNet flows are encapsulated/identified, and how functions of Packet Replication/Elimination/Ordering are implemented in an SRv6 domain. Congestion protection is also in the scope of this document.

Editor: This version only covers the functions of service protection and the congestion protection considerations will be added in the following versions.

5. SRv6 DetNet Data Plane Solution for Service Sub-layer

This section defines options of SRv6 data plane solution to support DetNet Service Sub-layer.

5.1. TLV Based SRv6 Data Plane Solution

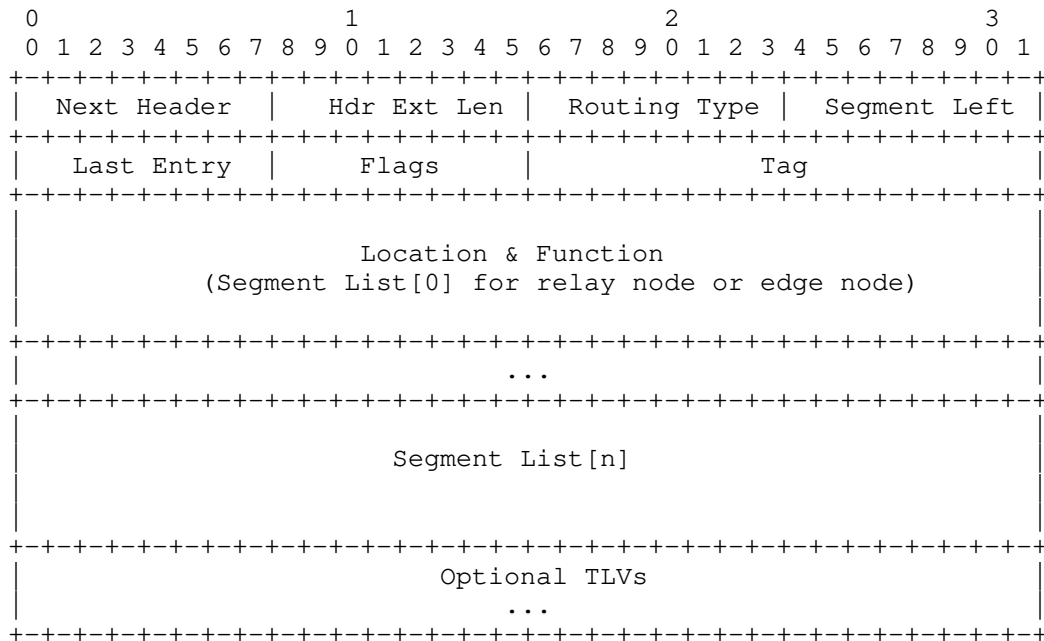
5.1.1. Encapsulation

An SRv6 Segment is a 128-bit value. SID is used as a shorter reference for "SRv6 Segment Identifier" or "SRv6 Segment". SRv6 SID can also be represented as LOC:FUNCT, where:

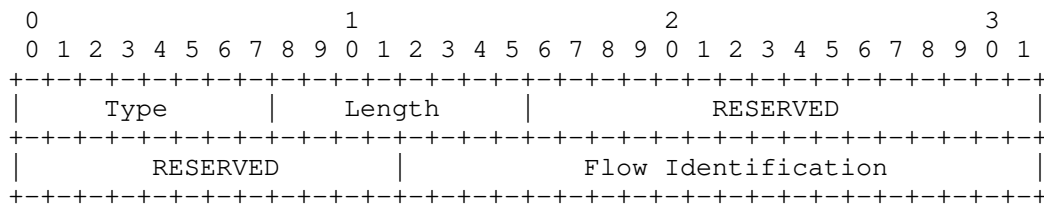
LOC, means "LOCATION" and defines the node associated with the SID (i.e.: represented by the SID).

FUNCT, means "FUNCTION", and identifies the processing that the node specified in LOC applies to the packet. See [I-D.filsfils-spring-srv6-network-programming] for details on SRv6 Network Programming.

The SRH for DetNet in the outer IPv6 header is showed as follows, according to [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming]:



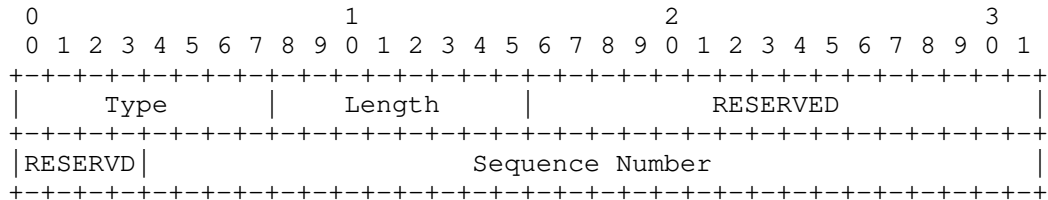
The SRH specification allows the use of optional TLVs. Two new TLVs are defined to support DetNet service protection. DetNet Flow Identification TLV is used to uniquely identify a DetNet flow in an SRv6 DetNet node. DetNet sequence number is used to discriminate packets in the same DetNet flow. They are defined as follows:



where:

- o Type: 8bits, to be assigned by IANA.
- o Length: 8 octets.

- o RESERVED: 28 bits, MUST be 0 on transmission and ignored on receipt.
- o Flow Identification: 20 bits, which is used for identifying DetNet flow.



where:

- o Type: 8 bits, to be assigned by IANA.
- o Length: 8.
- o RESERVED: 20 bits. MUST be 0 on transmission and ignored on receipt.
- o Sequence Number: 28 bits, which is used for indicating sequence number of a DetNet flow.

5.1.2. SRv6 Network Programming new Functions

New SRv6 Network Programming functions are defined as follows:

5.1.2.1. End. B.Replicationreserve the value of argument field(Inherited argument)of segment[0] of SRH n: Packet Replication Function

1. IF NH=SRH & SL>0 THEN
2. extract the DetNet TLV values from the SRH
3. create two new outer IPv6+SRH headers: IPv6-SRH-1 and IPv6-SRH-2
Insert the policy-instructed segment lists in each newly created SRH (SRH-1 and SRH-2). Also, add the extracted DetNet TLVs into SRH-1 and SRH-2.
4. remove the incoming outer IPv6+SRH header.
5. create a duplication of the incoming packet.
6. encapsulate the original packet into the first outer IPv6+SRH header: (IPv6-SRH-1) (original packet)

7. encapsulate the duplicate packet into the second outer IPv6+SRH header: (IPv6-SRH-2) (duplicate packet)
8. set the IPv6 SA as the local address of this node.
9. set the IPv6 DA of IPv6-SRH-1 to the first segment of the SRv6 Policy in of SRH-1 segment list.
10. set the IPv6 DA of IPv6-SRH-2 to the first segment of the SRv6 Policy in of SRH-2 segment list.
11. ELSE
12. drop the packet

5.1.2.2. End. B. Elimination: Packet Elimination Function

1. IF NH=SRH & SL>0 & "the packet is not a redundant packet" THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]
3. extract the value of DetNet TLVs from the SRH
4. create a new outer IPv6+SRH header
5. insert the policy-instructed segment lists in the newly created SRH and add the retrieved DetNet TLVs in the newly created SRH
6. remove the incoming outer IPv6+SRH header.
7. set the IPv6 DA to the first segment of the SRv6 Policy in the newly created SRH
8. ELSE
9. drop the packet

5.2. SID Based SRv6 Data Plane Solution

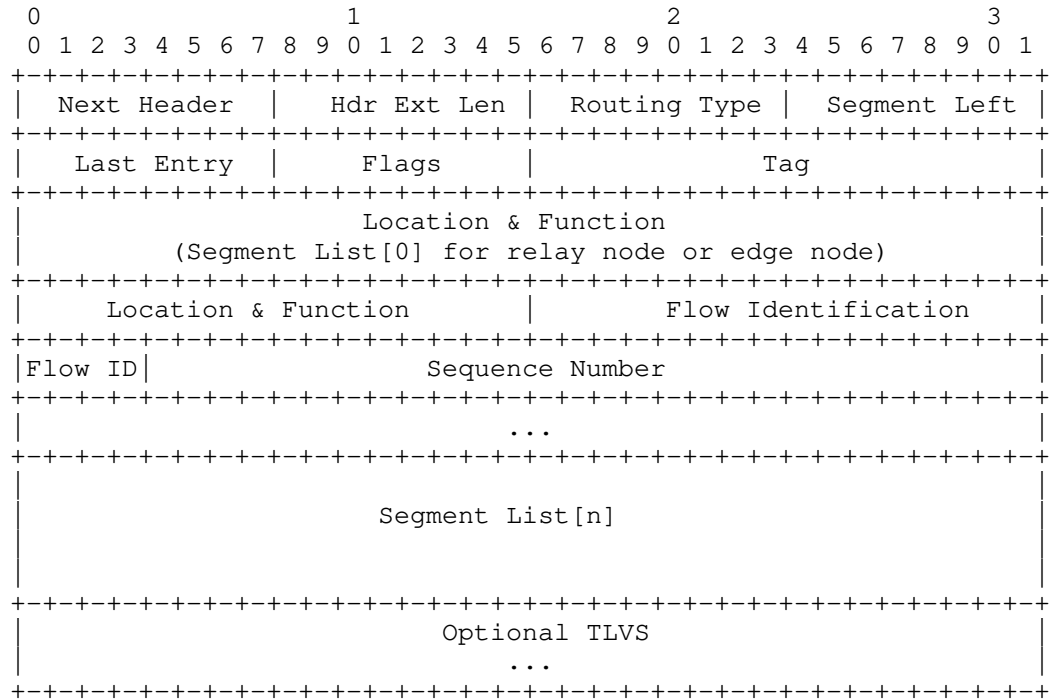
5.2.1. Encapsulation

SRv6 SID can be represented as LOC:FUNCT:ARG::, where:

LOC, means "LOCATION" and defines the node associated with the SID (i.e.: represented by the SID).

FUNCT, means "FUNCTION", and identifies the processing that the node specified in LOC applies to the packet.

ARG, means "ARGUMENTS" and provides the additional arguments for the function. New SID functions for DetNet is defined in section 5.2.2. See [I-D.filsfils-spring-srv6-network-programming] for details on SRV6 Network Programming. The SRH for DetNet in the outer IPv6 header is illustrated as follows



where:

- o LOCATION&FUNCTION: the 80 most significant bits that are used for routing the packet towards the LOCATION (as defined in [I-D.filsfils-spring-srv6-network-programming]);
- o FLOW IDENTIFICATION: 20 bits, in the DetNet TLVs in the SRH, used for DetNet flow identification in the DetNet relay node;
- o SEQUENCE NUMBER : 28 bits, in the DetNet TLVs, used for dis crime packets in the same DetNet flow;

5.2.2. Functions

New SID functions are defined as follows:

5.2.2.1. End. B.Replication: Packet Replication Function

The function is similar as that has been defined in section 5.1.2.1. The only difference is that instead of retrieving the TLV values, this function retrieves the argument.

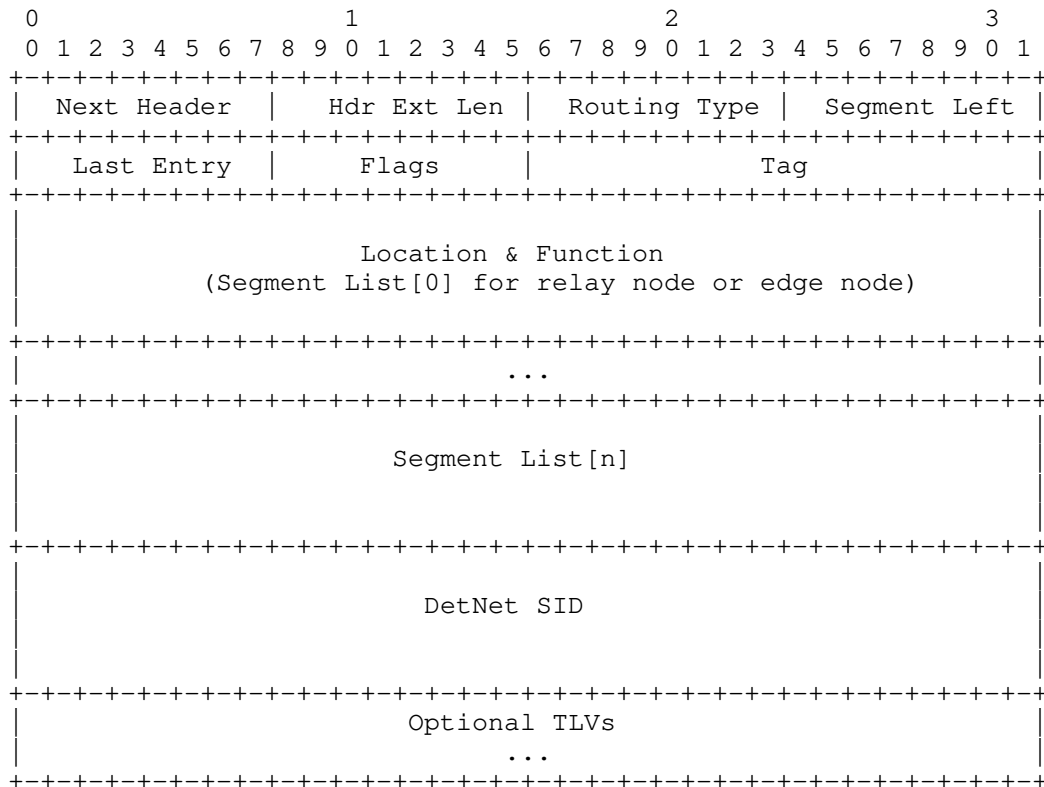
5.2.2.2. End. B. Elimination: Packet Elimination Function

The function is similar as that has been defined in section 5.1.2.2. The only difference is that instead of retrieving the TLV values, this function retrieves the argument.

5.3. DetNet SID Based SRv6 Data Plane Solution

5.3.1. Encapsulation

A non-forwarding DetNet SID is defined to carry Flow Identification and Sequence Number.



5.3.2. Functions

TBD

6. SRv6 DetNet Data Plane Solution for Transport Sub-layer

TBD

7. IANA Considerations

TBD

8. Security Considerations

TBD

9. Acknowledgements

Thank you for valuable comments from James Guichard and Andrew Mails.

10. Normative References

[I-D.filsfils-spring-srv6-network-programming]

Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.

[I-D.ietf-6man-segment-routing-header]

Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment
Routing Header (SRH)", draft-ietf-6man-segment-routing-
header-21 (work in progress), June 2019.

[I-D.ietf-detnet-architecture]

Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", draft-ietf-
detnet-architecture-13 (work in progress), May 2019.

[I-D.ietf-detnet-dp-sol-mpls]

Korhonen, J. and B. Varga, "DetNet MPLS Data Plane
Encapsulation", draft-ietf-detnet-dp-sol-mpls-02 (work in
progress), March 2019.

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d.,
bogdanov@google.com, b., and P. Mattes, "Segment Routing
Policy Architecture", draft-ietf-spring-segment-routing-
policy-03 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Yongqing Zhu
China Telecom

Email: zhuyq@gsta.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 13, 2020

X. Geng
M. Chen
Huawei
Y. Zhu
China Telecom
March 12, 2020

DetNet SRv6 Data Plane Encapsulation
draft-geng-detnet-dp-sol-srv6-02

Abstract

This document specifies Deterministic Networking data plane operation for SRv6 encapsulated user data.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 13, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Terminology and Conventions | 3 |
| 2.1. Terminology | 3 |
| 2.2. Conventions | 4 |
| 3. SRv6 DetNet Data Plane Overview | 4 |
| 3.1. SRv6 DetNet Data Plane Layers | 5 |
| 3.2. SRv6 DetNet Data Plane Scenarios | 5 |
| 4. SRv6 DetNet Data Plane Solution Considerations | 7 |
| 5. SRv6 DetNet Data Plane Solution for Service Sub-layer | 8 |
| 5.1. TLV Based SRv6 Data Plane Solution | 9 |
| 5.1.1. Encapsulation | 9 |
| 5.1.2. Replication SID and Elimination for DetNet | 11 |
| 5.2. SID Based SRv6 Data Plane Solution | 13 |
| 5.2.1. Encapsulation | 13 |
| 5.2.2. Functions | 14 |
| 5.3. DetNet SID Based SRv6 Data Plane Solution | 15 |
| 5.3.1. Encapsulation | 15 |
| 5.3.2. Functions | 15 |
| 6. SRv6 DetNet Data Plane Solution for Transport Sub-layer | 16 |
| 7. IANA Considerations | 16 |
| 8. Security Considerations | 16 |
| 9. Acknowledgements | 16 |
| 10. Normative References | 16 |
| Authors' Addresses | 17 |

1. Introduction

Deterministic Networking (DetNet), as described in [I-D.ietf-detnet-architecture] provides a capability to carry specified data flows with extremely low data loss rates and bounded latency within a network domain. DetNet is enabled by a group of technologies, such as resource allocation, service protection and explicit routes.

Segment Routing(SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". SR can be applied over IPv6 data plane using the Segment Routing Extension Header (SRH, [I-D.ietf-6man-segment-routing-header]). A segment in segment routing terminology is not limited to a routing/forwarding function.

A segment can be associated to an arbitrary processing of the packet in the node identified by the segment. In other words, an SRv6 Segment can indicate functions that are executed locally in the node where they are defined. SRv6 network Programming [I-D.filsfils-spring-srv6-network-programming] describe the different segments and functions associated to them.

This document describes how to implement DetNet in an SRv6 enabled domain, including :

- o Source routing, which steers the DetNet flows through the network according to an explicit path with allocated resources;
- o Network programming, which applies instructions (functions) to packets in some special nodes (or even all the nodes) along the path in order to guarantee, e.g., service protection and congestion protection.

DetNet SRv6 encapsulation and new SRv6 functions ([I-D.filsfils-spring-srv6-network-programming]) for DetNet are defined in this document. Control plane and OAM are not in the scope of this document.

Control plane and OAM are not in the scope of this document.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1. Terminology

Terminologies for DetNet go along with the definition in [I-D.ietf-detnet-architecture] and [RFC8402]. Other terminologies are defined as follows:

- o NH: The IPv6 next-header field.
- o SID: A Segment Identifier ([RFC8402]).
- o SRH: The Segment Routing Header ([I-D.ietf-6man-segment-routing-header]).

2.2. Conventions

Conventions in the document are defined as follows:

- o NH=SRH means that NH is 43 with routing type 4 which is (as defined in [I-D.ietf-6man-segment-routing-header]), the values representing the SRH.
- o A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.
- o SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3. It has to be noted that [I-D.ietf-6man-segment-routing-header] defines the segment list encoding in the reverse order of the path. A path represented by <S1,S2,S3>, will be encoded in the SRH as follows:

SegmentList[0]=S3

SegmentList[1]=S2

SegmentList[2]=S1

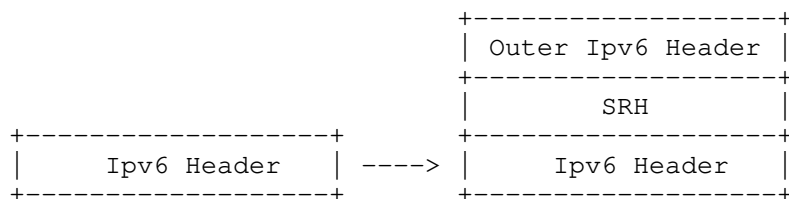
The reverse encoding has been defined in order to optimise the processing time of the segment list. See [draft-ietf-6man-segment-routing-header] for more details.

- o (SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:
 - IPv6 header with source and destination addresses SA and DA respectively, and next-header set to SRH (i.e.: 43 with type 4), with a list of segments(SIDs) <S1, S2, S3> with SegmentsLeft = SL
 - The payload of the packet is not represented
 - (S3, S2, S1; SL) represents the same SID list as <S1, S2, S3>, but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID

3. SRv6 DetNet Data Plane Overview

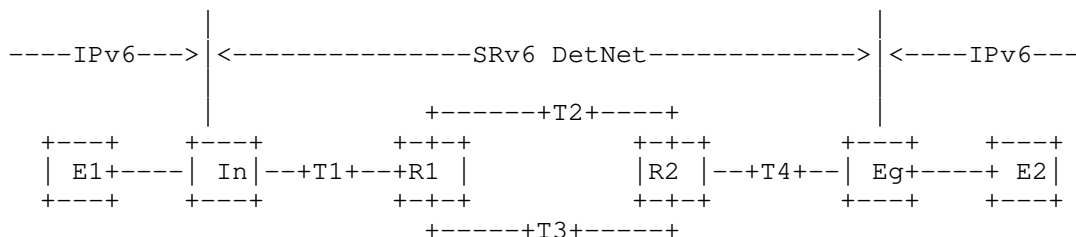
3.1. SRv6 DetNet Data Plane Layers

[I-D.ietf-detnet-architecture]decomposes the DetNet data plane into two sub-layers: service sub-layer and transport sub-layer. Different from DetNet MPLS data plane solution, which uses DetNet Control Word(d-CW) and S-Label to support service sub-layer and uses T-Label to support transport sub-layer, no explicit sub-layer division exists in SRv6 data plane. A classical SRv6 DetNet data plane solution is showed in the picture below:



The outer IPv6 Header with the SRH is used for carrying DetNet flows. Traffic Engineering is instantiated in the segment list of SRH, and other functions and arguments for service protection (packet replication, elimination and ordering) and congestion control (packet queuing and forwarding) are also defined in the SRH.

3.2. SRv6 DetNet Data Plane Scenarios



The figure above shows that an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the outer IPv6 header+SRH of the packet is popped, and the packet is sent to the destination E2.

The figure above shows that an IPv6 flow is sent our from the end station: E1. The packet of the flow is encapsulated as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the upper IPv6 header with SRH of the packet is popped, and the packet is transmitted to the destination(E2).

The DetNet packet processing is as follows:

Ingress:

Inserts the SRv6 Policy that will steer the packet from Ingress to the destination

The methods and mechanisms used for defining, instantiating and applying the policy are outside of this document. An example of policies are described in [I-D.ietf-spring-segment-routing-policy]

Flow Identification and Sequence Number are carried in the SRH optionally.

Relay Node 1(Replication Node):

Replicates the payload and IPv6 Header with the SRH. This is a new function in the context of SRv6 Network Programming which will associate a given SID to a replication instruction in the node originating and advertising the SID. The replication instruction includes:

- * The removal of the existing IPv6+SRH header
- * The encapsulation into a new outer IPv6+SRH header. Each packet (the original and the duplicated) are encapsulated into respectively new outer IPv6+SRH headers.

Binding two different SRv6 Policies respectively to the original packet and the replicated packet, which can steer the packets from Relay Node 1 to Relay Node 2 through two tunnels.

If Flow Identification and Sequence Number are not carried in the SRH in the ingress, add it them in the SRH.

Relay Node 2(Elimination Node):

Eliminates the redundant packets.

Binds a new SRv6 Policy to the survival packet, which steers the packet from Relay Node 2 to Egress.

Egress:

Decapsulates the outer Ipv6 header.

Sends the inter packet to the End Station 2.

The DetNet packet encapsulation is illustrated here below. It has to be noted that, in the example below, the R2 address is a SRH SID associated to a TBD function related to the packet replication the node R1 has to perform. The same (or reverse) apply to node R2 which is in charge of the discard of the duplicated packet. Here also a new function will have a new SID allocated to it and representing the delete of the duplication in R2.

End Station1 output packet: (E1,E2)

Ingress output packet: (In, T1) (R1,T1, SL=2) (E1,E2)

Transit Node1 output packet: (In, R1) (R1,T1,SL=1) (E1,E2)

Relay Node1 output packets : (R1,T2) (R2,T2,SL=2) (E1,E2),
(R1,T3) (R2,T3,SL=2) (E1,E2)

Transit Node2 output packet: (R1, R2) (R2,T2,SL=1) (E1,E2)

Transit Node3 output packet: (R1, R2) (R2,T3,SL=1) (E1,E2)

Relay Node2 output packet: (R2, T4) (Eg,T4,SL=2) (E1,E2)

Transit Node4 output packet: (R2, Eg) (Eg,T4,SL=1) (E1,E2)

Egress out : (E1,E2)

4. SRv6 DetNet Data Plane Solution Considerations

To carry DetNet over SRv6, the following elements are required:

1. A method of identifying the SRv6 payload type;
2. A suitable explicit path to deliver the DetNet flow ;
3. A method of indicating packet processing, such as PREOF(Packet Replication, Elimination and Ordering as defined in [I-D.ietf-detnet-architecture]);
4. A method of identifying the DetNet flow;
5. A method of carrying DetNet sequence number;
6. A method of carrying queuing and forwarding indication to do congestion protection;

In this design, DetNet flows are encapsulated in an outer IPv6+SRH header at the Ingress Node. The SR policy identified in the SRH steers the DetNet flow along a selected path. The explicit path followed by a DetNet flow, which protect it from temporary interruptions caused by the convergence of routing, is encoded within the SID list of the SR policy. The network device inside the DetNet domain forwards the packet according to IPv6 Destination Address(DA), and the IPv6 DA is updated with the SID List according to SRv6 forwarding procedures defined in [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming]

With SRv6 network programming, the SID list can also give instruments representing a function to be called at the node in the DetNet domain. Therefore DetNet specific functions defined in [I-D.ietf-detnet-architecture], corresponding to local packet processing in the network, can also be implemented by SRv6. New functions associated with SIDs for DetNet are defined in this document.

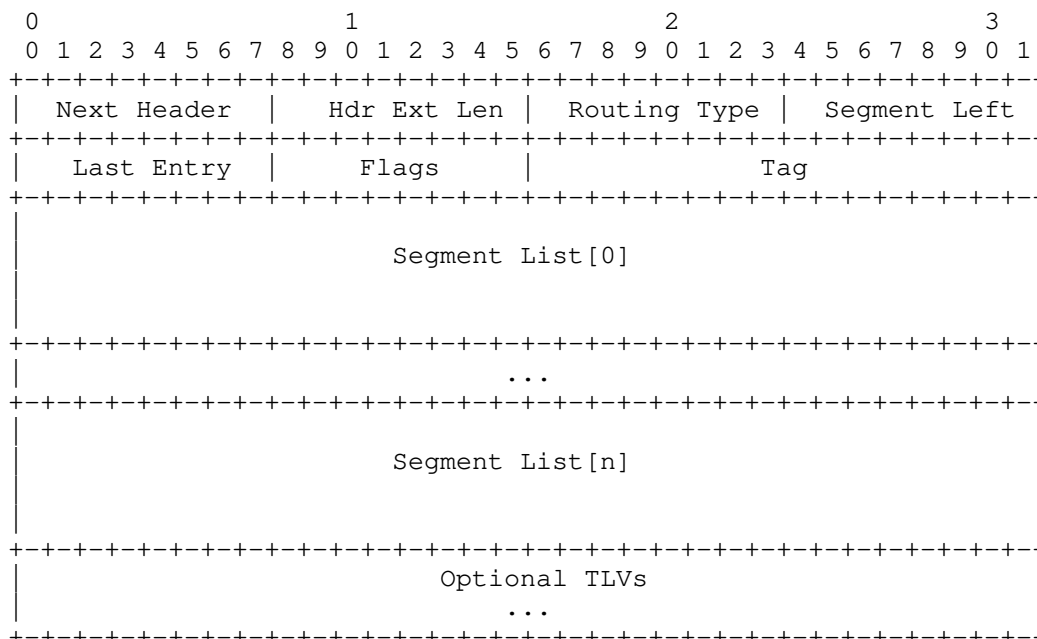
This document describes how DetNet flows are encapsulated/identified, and how functions of Packet Replication/Elimination/Ordering are implemented in an SRv6 domain. Congestion protection is also in the scope of this document.

Editor: This version only covers the functions of service protection and the congestion protection considerations will be added in the following versions.

5. SRv6 DetNet Data Plane Solution for Service Sub-layer

This section defines options of SRv6 data plane solution to support DetNet Service Sub-layer.

SRH is as follows, which defined in [I-D.ietf-6man-segment-routing-header]



The SRH specification allows the use of optional TLVs.

Each SRv6 Segment in the segment list is a 128-bit value. SID is used as a shorter reference for "SRv6 Segment Identifier" or "SRV6 Segment". SRv6 SID can also be represented as LOC:FUNCT, where:

LOC, means "LOCATION" and defines the node associated with the SID (i.e.: represented by the SID).

FUNCT, means "FUNCTION", and identifies the processing that the node specified in LOC applies to the packet. See [I-D.filsfils-spring-srv6-network-programming] for details on SRV6 Network Programming.

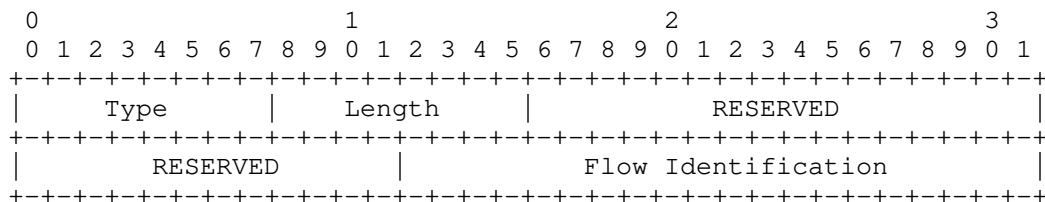
as defined in [I-D.filsfils-spring-srv6-network-programming].

5.1. TLV Based SRv6 Data Plane Solution

5.1.1. Encapsulation

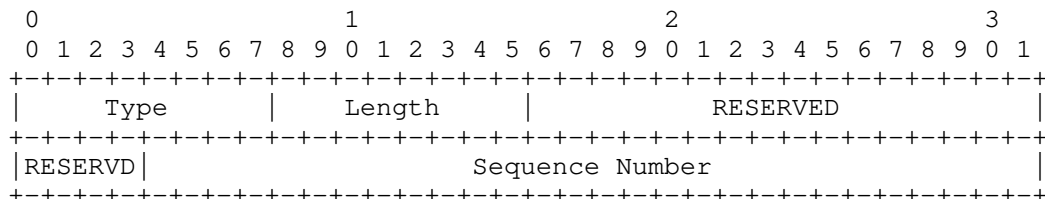
Two new TLVs are defined to support DetNet service protection. DetNet Flow Identification TLV is used to uniquely identify a DetNet flow in an SRv6 DetNet node. DetNet sequence number is used to discriminate packets in the same DetNet flow. They are defined as follows:

Option 1: separated TLVs for flow identification and sequence number



where:

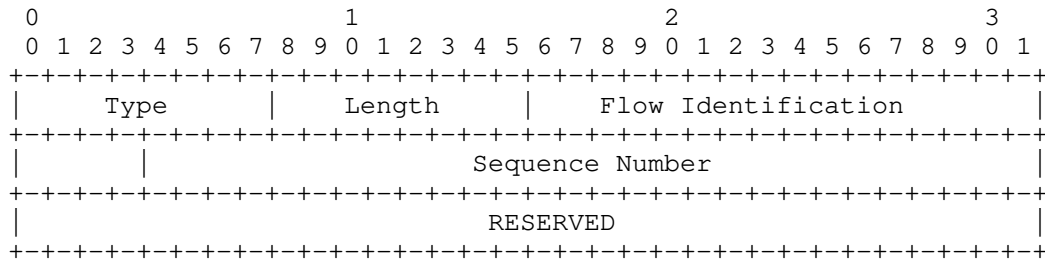
- o Type: 8bits, to be assigned by IANA.
- o Length: 8 octets.
- o RESERVED: 28 bits, MUST be 0 on transmission and ignored on receipt.
- o Flow Identification: 20 bits, which is used for identifying DetNet flow.



where:

- o Type: 8 bits, to be assigned by IANA.
- o Length: 8.
- o RESERVED: 20 bits. MUST be 0 on transmission and ignored on receipt.
- o Sequence Number: 28 bits, which is used for indicating sequence number of a DetNet flow.

Option 2: unified TLV for flow identification and sequence number



where:

- o Type: 8 bits, to be assigned by IANA.
- o Length: 12.
- o Flow Identification: 20 bits, which is used for identifying DetNet flow.
- o Sequence Number: 28 bits, which is used for indicating sequence number of a DetNet flow.
- o RESERVED: 32 bits. MUST be 0 on transmission and ignored on receipt.

5.1.2. Replication SID and Elimination for DetNet

New SIDs, replication SID and elimination SID, are defined as follows:

5.1.2.1. Replication SID

Redundancy SID is a variation of binding SID defined in [I-D.ietf-spring-segment-routing-policy] Redundancy SID indicates the following operations:

- o Steering the packet into the corresponding redundancy policy
- o Packet replication based on the redundancy policy, e.g., the number of replication copies
- o Encapsulate the packet with necessary meta data (e.g., flow identification, sequence number) if it is not included in the original packet

When N receives a packet whose IPv6 DA is S and S is a Replication SID, N could do:

1. IF NH=SRH & SL>0 THEN
2. extract the DetNet TLV values from the SRH
3. create two new outer IPv6+SRH headers: IPv6-SRH-1 and IPv6-SRH-2
Insert the policy-instructed segment lists in each newly created
SRH (SRH-1 and SRH-2). Also, add the extracted DetNet TLVs into
SRH-1 and SRH-2.
4. remove the incoming outer IPv6+SRH header.
5. create a duplication of the incoming packet.
6. encapsulate the original packet into the first outer IPv6+SRH
header: (IPv6-SRH-1) (original packet)
7. encapsulate the duplicate packet into the second outer IPv6+SRH
header: (IPv6-SRH-2) (duplicate packet)
8. set the IPv6 SA as the local address of this node.
9. set the IPv6 DA of IPv6-SRH-1 to the first segment of the SRv6
Policy in of SRH-1 segment list.
10. set the IPv6 DA of IPv6-SRH-2 to the first segment of the SRv6
Policy in of SRH-2 segment list.
11. ELSE
12. drop the packet

5.1.2.2. Elimination SID

Elimination SID indicates the following operations:

- o Packet elimination: forward the first received packets and
eliminate the redundant packets.
- o Packet ordering(optional): reorder the packets if the packet
arrives out of order

When N receives a packet whose IPv6 DA is S and S is a Elimination
SID, N could do:

1. IF NH=SRH & SL>0 & "the packet is not a redundant packet" THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]

3. extract the value of DetNet TLVs from the SRH
4. create a new outer IPv6+SRH header
5. insert the policy-instructed segment lists in the newly created SRH and add the retrieved DetNet TLVs in the newly created SRH
6. remove the incoming outer IPv6+SRH header.
7. set the IPv6 DA to the first segment of the SRv6 Policy in the newly created SRH
8. ELSE
9. drop the packet

5.2. SID Based SRv6 Data Plane Solution

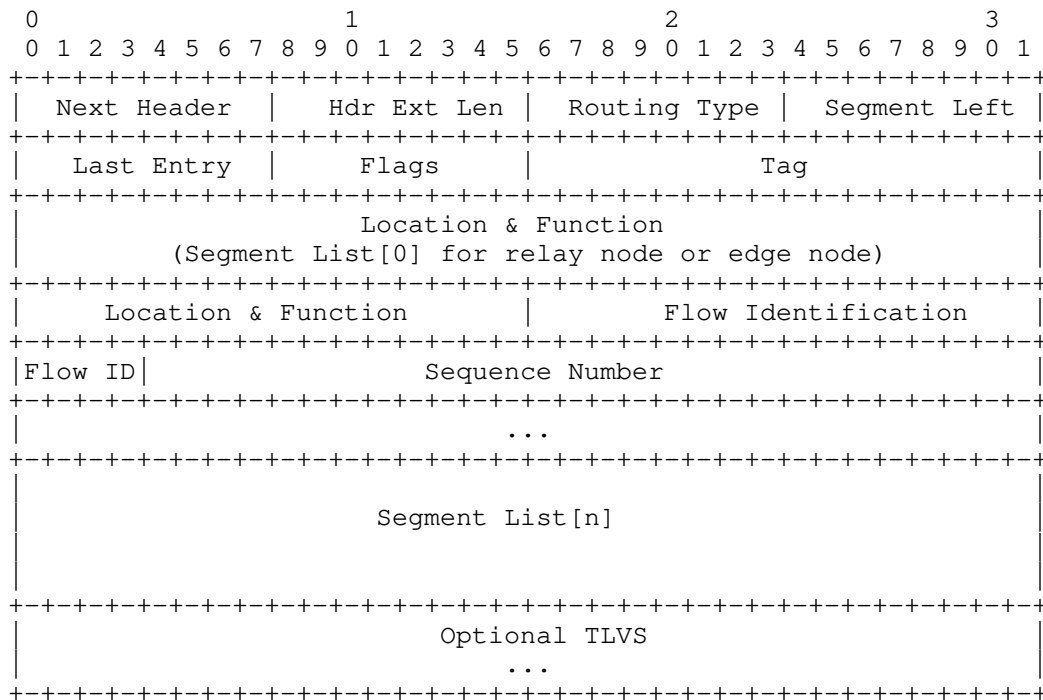
5.2.1. Encapsulation

SRv6 SID can be represented as LOC:FUNCT:ARG::, where:

LOC, means "LOCATION" and defines the node associated with the SID (i.e.: represented by the SID).

FUNCT, means "FUNCTION", and identifies the processing that the node specified in LOC applies to the packet.

ARG, means "ARGUMENTS" and provides the additional arguments for the function. New SID functions for DetNet is defined in section 5.2.2. See [I-D.filsfils-spring-srv6-network-programming] for details on SRV6 Network Programming. The SRH for DetNet in the outer IPv6 header is illustrated as follows



where:

- o LOCATION&FUNCTION: the 80 most significant bits that are used for routing the packet towards the LOCATION (as defined in [I-D.filsfils-spring-srv6-network-programming]);
- o FLOW IDENTIFICATION: 20 bits, in the DetNet TLVs in the SRH, used for DetNet flow identification in the DetNet relay node;
- o SEQUENCE NUMBER : 28 bits, in the DetNet TLVs, used for dis crime packets in the same DetNet flow;

5.2.2. Functions

New SID functions are defined as follows:

5.2.2.1. End. B.Replication: Packet Replication Function

The function is similar as that has been defined in section 5.1.2.1. The only difference is that instead of retrieving the TLV values, this function retrieves the argument.

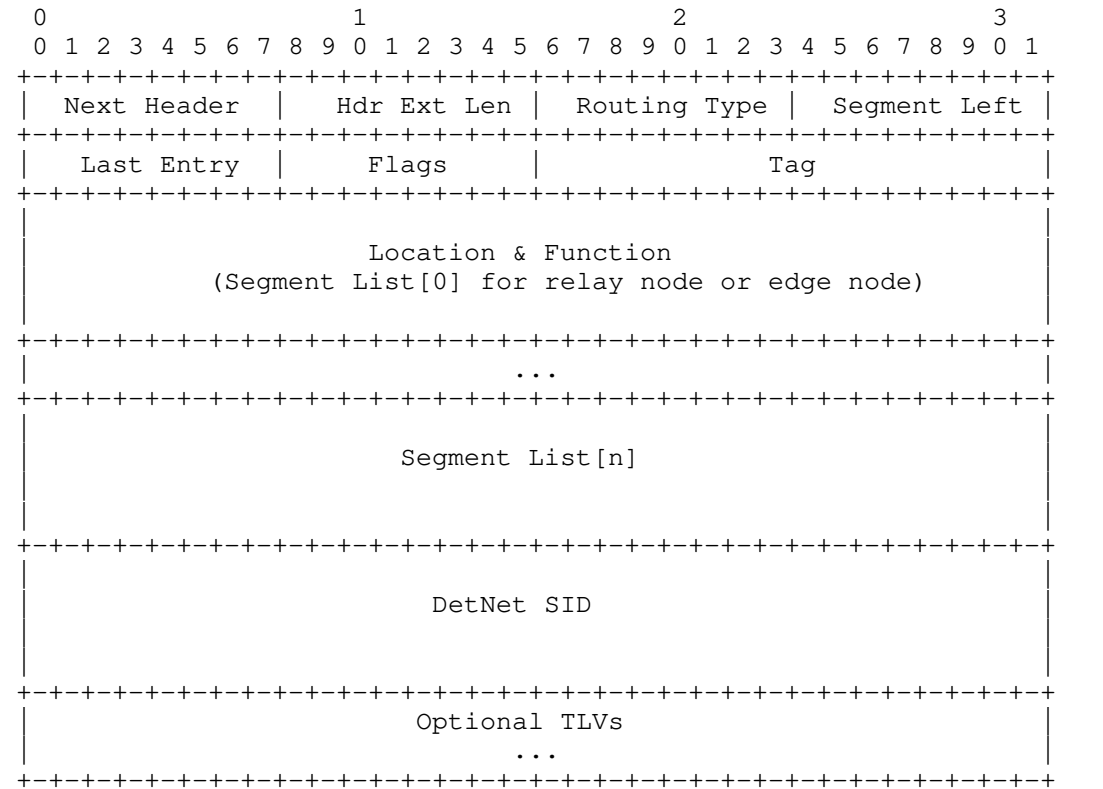
5.2.2.2. End. B. Elimination: Packet Elimination Function

The function is similar as that has been defined in section 5.1.2.2. The only difference is that instead of retrieving the TLV values, this function retrieves the argument.

5.3. DetNet SID Based SRv6 Data Plane Solution

5.3.1. Encapsulation

A non-forwarding DetNet SID is defined to carry Flow Identification and Sequence Number.



5.3.2. Functions

TBD

6. SRv6 DetNet Data Plane Solution for Transport Sub-layer

TBD

7. IANA Considerations

TBD

8. Security Considerations

TBD

9. Acknowledgements

Thank you for valuable comments from James Guichard and Andrew Mails.

10. Normative References

[I-D.filsfils-spring-srv6-network-programming]

Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.

[I-D.ietf-6man-segment-routing-header]

Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header
(SRH)", draft-ietf-6man-segment-routing-header-26 (work in
progress), October 2019.

[I-D.ietf-detnet-architecture]

Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", draft-ietf-
detnet-architecture-13 (work in progress), May 2019.

[I-D.ietf-detnet-dp-sol-mpls]

Korhonen, J. and B. Varga, "DetNet MPLS Data Plane
Encapsulation", draft-ietf-detnet-dp-sol-mpls-02 (work in
progress), March 2019.

[I-D.ietf-spring-segment-routing-policy]

Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and
P. Mattes, "Segment Routing Policy Architecture", draft-
ietf-spring-segment-routing-policy-06 (work in progress),
December 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Yongqing Zhu
China Telecom

Email: zhuyq@gsta.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

X. Geng
Z. Li
M. Chen
Huawei
July 8, 2019

SRv6 for Deterministic Networking (DetNet)
draft-geng-spring-srv6-for-detnet-00

Abstract

Deterministic Networking provides service with low jitter, bounded latency and low loss rate, using technologies of explicit route, resource reservation and service protection. This document specifies how to implement Deterministic Networking (DetNet) in a SRv6 Network.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. SRv6 for DetNet Overview | 4 |
| 4. Service Protection | 5 |
| 4.1. Service Protection Scenarios | 6 |
| 4.2. Data Plane Considerations | 8 |
| 4.3. Control Plane Considerations | 8 |
| 4.4. Functions for Service Protection | 9 |
| 4.4.1. End. B. Replication: Packet Replication Function . . | 9 |
| 4.4.2. End. B. Elimination: Packet Elimination Function . . | 9 |
| 5. Explicit Route | 10 |
| 6. IANA Considerations | 10 |
| 7. Security Considerations | 10 |
| 8. Acknowledgements | 10 |
| 9. Normative References | 10 |
| Authors' Addresses | 11 |

1. Introduction

With more and more applications running in the Internet, quality of the service gains more and more attention, especially for some new applications, for example Cloud VR, Cloud Game, HDV (high-definition video) and so on, SLA (Service Level Agreement), including jitter, delay and loss rate, influences the users' experience significantly. So SLA guarantee is an important issue which requires new technologies and solutions for the network.

Deterministic Networking (DetNet defined in [I-D.ietf-detnet-architecture]) provides a capability to carry specified data flows for real-time applications with extremely low data loss rates, low jitter and bounded latency within a network domain. Techniques used include: 1) providing explicit paths for DetNet flows that satisfies the SLA requirement from user and do not immediately change with the network topology; 2) reserving data plane resources for DetNet flows along the allocated path of the flow; 3) replicates DetNet flows into two or more copies and transport different copies through different path in parallel, called service protection.

Segment Routing (SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". SR can be applied over IPv6 data plane using Routing Extension Header (SRH, [I-D.ietf-6man-segment-routing-header]). A segment in Segment Routing is not limited to a routing/forwarding function. An SRv6 Segment can indicate functions that are executed locally in the node where they are defined.

[I-D.filsfils-spring-srv6-network-programming] describes some well-known functions and segments associated to them. SRH TLVs ([I-D.ietf-6man-segment-routing-header]) also provides meta-data for segment processing. All these features make SRv6 suitable to carry DetNet flows, by defining new segments associated with DetNet functions and meta data for DetNet.

This document describes the concepts needed by implementing DetNet in an SRv6-based domain and provides considerations for any corresponding implementation.

Editor's note:

1. DetNet is limited in a controlled network domain, and it is not the only method to provide SLA guarantee. But it is a good start to discuss how to use SRv6 to have a SLA-guaranteed network service.

2. Resource Reservation will be added in future work.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terminologies for DetNet go along with the definition in [I-D.ietf-detnet-architecture] and [I-D.ietf-6man-segment-routing-header]:

DetNet domain

The portion of a network that is DetNet aware. It includes end systems and DetNet nodes

DetNet edge node

An instance of a DetNet relay node that acts as a source and/ or destination at the DetNet service sub-layer. For example, it can include a DetNet service sub-layer proxy function for DetNet service protection (e.g., the addition or removal of packet

sequencing information) for one or more end systems, or starts or terminates resource allocation at the DetNet forwarding sub-layer, or aggregates DetNet services into new DetNet flows. It is analogous to a Label Edge Router (LER) or a Provider Edge (PE) router.

DetNet relay node

A DetNet node including a service sub-layer function that interconnects different DetNet forwarding sub-layer paths to provide service protection. A DetNet relay node participates in the DetNet service sub-layer. It typically incorporates DetNet forwarding sub-layer functions as well, in which case it is collocated with a transit node.

DetNet transit node

A DetNet node operating at the DetNet forwarding sub-layer, that utilizes link layer and/or network layer switching across multiple links and/or sub-networks to provide paths for DetNet service sub-layer functions. Typically provides resource allocation over those paths. An MPLS LSR is an example of a DetNet transit node.

End system

End systems of interest to this document are either sources or destinations of DetNet flows. An end system may or may not be DetNet forwarding sub-layer aware or DetNet service sub-layer aware.

DetNet service sub-layer

DetNet functionality is divided into two sub-layers. One of them is the DetNet service sub-layer, at which a DetNet service, e.g., service protection is provided.

DetNet forwarding sub-layer

DetNet functionality is divided into two sub-layers. One of them is the DetNet forwarding sub-layer, which optionally provides resource allocation for DetNet flows over paths provided by the underlying network.

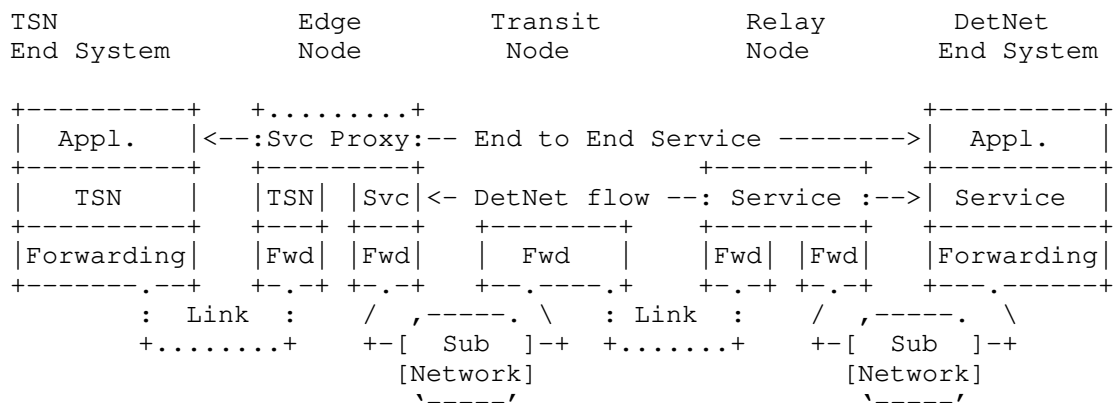
3. SRv6 for DetNet Overview

As mentioned above, there are mainly three technologies/functions defined in DetNet: Explicit Route, Resource Reservation and Service Protection. Explicit Route is the basis of the other two

technologies, and guarantees the path satisfies the SLA requirement from application. Resource Reservation protects DetNet flows from network congestion, which could reduce the end-to-end latency and congestion loss; Service Protection prevents DetNet flow from random media errors and equipment failures, which makes the loss rate extremely low.

In [I-D.ietf-detnet-architecture], DetNet functionality is implemented in two sub-layers in the protocol stack: the DetNet service sub-layer and the DetNet forwarding sub-layer. The DetNet service sub-layer provides DetNet service, e.g., service protection. The DetNet forwarding sub-layer supports DetNet service in the underlying network, by providing explicit routes and resource allocation to DetNet flows. There is no obvious protocol stack as MPLS, in SRv6 both service sub-layer and forwarding sub-layer are implemented through SRH.

The following picture shows that a general DetNet enabled network defined in [I-D.ietf-detnet-architecture] :

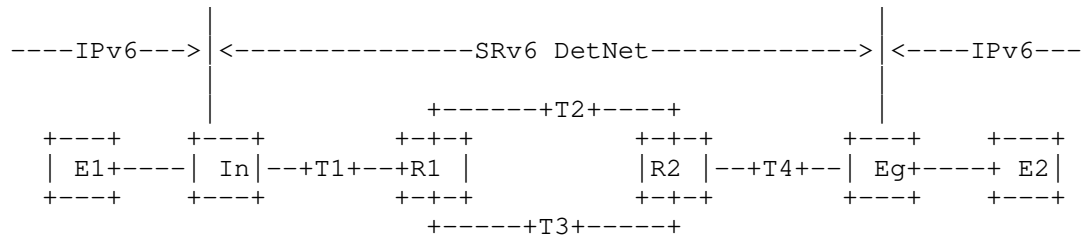


In SRv6 for DetNet, the outer IPv6 Header with the SRH is used for carrying DetNet flows. Explicit path is instantiated in the segment list of SRH, and other functions/arguments for service protection (packet replication, elimination and ordering, PREOF) and resource reservation (packet queuing and scheduling) are also defined in the specified SID. Meta data for DetNet could be instantiated in the Optional TLVs.

4. Service Protection

4.1. Service Protection Scenarios

The figure below shows that an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the outer IPv6 header+SRH of the packet is popped, and the packet is sent to the destination E2.



The DetNet packet processing is as follows:

Ingress:

Inserts the SRv6 Policy that will steer the packet from Ingress to the destination

The methods and mechanisms used for defining, instantiating and applying the policy are outside of this document. An example of policies are described in [I-D.ietf-spring-segment-routing-policy]

Flow Identification and Sequence Number are carried in the SRH.

Relay Node 1(Replication Node):

Replicates the payload and IPv6 Header with the SRH. This is a new function in the context of SRv6 Network Programming which will associate a given SID to a replication instruction in the node originating and advertising the SID. The replication instruction includes:

- * The removal of the existing IPv6+SRH header
- * The encapsulation into a new outer IPv6+SRH header. Each packet (the original and the duplicated) are encapsulated into respectively new outer IPv6+SRH headers.

Binding two different SRv6 Policies respectively to the original packet and the replicated packet, which can steer the packets from Relay Node 1 to Relay Node 2 through two tunnels.

Relay Node 2 (Elimination Node):

Eliminates the redundant packets.

Binds a new SRv6 Policy to the survival packet, which steers the packet from Relay Node 2 to Egress.

Egress:

Decapsulates the outer Ipv6 header.

Sends the inter packet to the End Station 2.

The DetNet packet encapsulation is illustrated here below. It has to be noted that, in the example below, the R2 address is a SRH SID associated to a TBD function related to the packet replication the node R1 has to perform. The same (or reverse) apply to node R2 which is in charge of the discard of the duplicated packet. Here also a new function will have a new SID allocated to it and representing the delete of the duplication in R2.

End Station1 output packet: (E1,E2)

Ingress output packet: (In, T1) (R1,T1, SL=2) (E1,E2)

Transit Node1 output packet: (In, R1) (R1,T1,SL=1) (E1,E2)

Relay Node1 output packets : (R1,T2) (R2,T2,SL=2) (E1,E2),
(R1,T3) (R2,T3,SL=2) (E1,E2)

Transit Node2 output packet: (R1, R2) (R2,T2,SL=1) (E1,E2)

Transit Node3 output packet: (R1, R2) (R2,T3,SL=1) (E1,E2)

Relay Node2 output packet: (R2, T4) (Eg,T4,SL=2) (E1,E2)

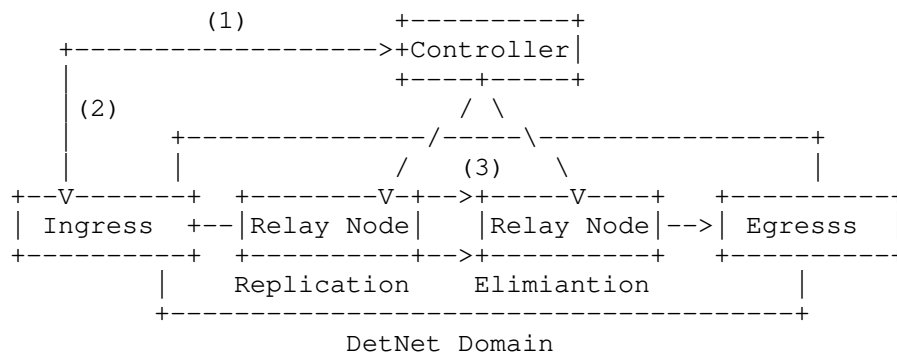
Transit Node4 output packet: (R2, Eg) (Eg,T4,SL=1) (E1,E2)

Egress out : (E1,E2)

4.2. Data Plane Considerations

Flow Identification and sequence number are necessary in the encapsulation of SRv6 for DetNet in order to support service protection. Replication nodes decide which DetNet flows are supposed to be replicated by identifying the flow with the flow identification. Elimination nodes decide whether a packet should be dropped because of redundancy by identifying the flow identification and sequence number.

4.3. Control Plane Considerations



1. Edge node->Controller: Sends a path computation requirement containing that service protection in order to have ultra-reliability through PCEP/BGP extensions.

2. Controller->Edge node: Computes a P2MP2P path, including replication nodes and elimination nodes. Between a pair of replication node and elimination node, there are more than one path, and if any equipment failure happens in one of them, the DetNet service is not interrupted. Send the path computation result to the edge node through PCEP/BGP extensions.

3. Controller->Relay Node : In a P2MP2P path, every pair of replication node and elimination node should be configured to identify different DetNet flows by the different flow identifications, and the rule of sequence number should be negotiated between relay nodes. After replication or elimination, the explicit path to the next relay is also required through BGP extensions or Netconf YANG.

4.4. Functions for Service Protection

New SRv6 Network Programming functions are defined as follows:

4.4.1. End. B. Replication: Packet Replication Function

1. IF NH=SRH & SL>0 THEN
2. extract the DetNet TLV values from the SRH
3. create two new outer IPv6+SRH headers: IPv6-SRH-1 and IPv6-SRH-2
Insert the policy-instructed segment lists in each newly created SRH (SRH-1 and SRH-2). Also, add the extracted DetNet TLVs into SRH-1 and SRH-2.
4. remove the incoming outer IPv6+SRH header.
5. create a duplication of the incoming packet.
6. encapsulate the original packet into the first outer IPv6+SRH header: (IPv6-SRH-1) (original packet)
7. encapsulate the duplicate packet into the second outer IPv6+SRH header: (IPv6-SRH-2) (duplicate packet)
8. set the IPv6 SA as the local address of this node.
9. set the IPv6 DA of IPv6-SRH-1 to the first segment of the SRv6 Policy in of SRH-1 segment list.
10. set the IPv6 DA of IPv6-SRH-2 to the first segment of the SRv6 Policy in of SRH-2 segment list.
11. ELSE
12. drop the packet

4.4.2. End. B. Elimination: Packet Elimination Function

1. IF NH=SRH & SL>0 & "the packet is not a redundant packet" THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]
3. extract the value of DetNet TLVs from the SRH
4. create a new outer IPv6+SRH header

5. insert the policy-instructed segment lists in the newly created SRH and add the retrieved DetNet TLVs in the newly created SRH
6. remove the incoming outer IPv6+SRH header.
7. set the IPv6 DA to the first segment of the SRv6 Policy in the newly created SRH
8. ELSE
9. drop the packet

5. Explicit Route

SRv6 could support explicit route using segment list without extra extension. In DetNet, explicit route could be used with service protection or resource reservation. When explicit route works with service protection, a P2MP2P path is required to indicate the behavior of replication and elimination. When explicit route works with resource reservation, the explicit path indicates the nodes or links a DetNet flow goes through, and also indicates the resource allocated for the DetNet flow in the specified nodes or links.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

TBD

8. Acknowledgements

Thank you for valuable comments from James Guichard and Andrew Mails.

9. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment
Routing Header (SRH)", draft-ietf-6man-segment-routing-
header-21 (work in progress), June 2019.
- [I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", draft-ietf-
detnet-architecture-13 (work in progress), May 2019.
- [I-D.ietf-detnet-dp-sol-mpls]
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane
Encapsulation", draft-ietf-detnet-dp-sol-mpls-02 (work in
progress), March 2019.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d.,
bogdanov@google.com, b., and P. Mattes, "Segment Routing
Policy Architecture", draft-ietf-spring-segment-routing-
policy-03 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 14, 2021

X. Geng
Z. Li
M. Chen
Huawei
July 13, 2020

SRv6 for Deterministic Networking (DetNet)
draft-geng-spring-srv6-for-detnet-01

Abstract

Deterministic Networking provides service with low jitter, bounded latency and low loss rate, using technologies of explicit route, resource reservation and service protection. This document specifies how to implement Deterministic Networking (DetNet) in a SRv6 Network.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. SRv6 for DetNet Overview | 4 |
| 4. Service Protection | 5 |
| 4.1. Service Protection Scenarios | 5 |
| 4.2. Data Plane Considerations | 7 |
| 4.3. Control Plane Considerations | 7 |
| 5. Resource Reservation | 8 |
| 5.1. Resource Reservation Scenarios | 8 |
| 5.2. Data Plane Considerations | 10 |
| 5.3. Control Plane Considerations | 10 |
| 6. Explicit Route | 11 |
| 7. IANA Considerations | 11 |
| 8. Security Considerations | 11 |
| 9. Acknowledgements | 11 |
| 10. Normative References | 11 |
| Authors' Addresses | 12 |

1. Introduction

With more and more applications running in the Internet, quality of the service gains more and more attention, especially for some new applications, for example Cloud VR, Cloud Game, HDV (high-definition video) and so on, SLA (Service Level Agreement), including jitter, delay and loss rate, influences the users' experience significantly. So SLA guarantee is an important issue which requires new technologies and solutions for the network.

Deterministic Networking (DetNet defined in [RFC8655]) provides a capability to carry specified data flows for real-time applications with extremely low data loss rates, low jitter and bounded latency within a network domain. Techniques used include: 1) providing explicit paths for DetNet flows that satisfies the SLA requirement from user and do not immediately change with the network topology; 2) reserving data plane resources for DetNet flows along the allocated path of the flow; 3) replicates DetNet flows into two or more copies and transport different copies through different path in parallel, called service protection.

Segment Routing(SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". SR can be applied over IPv6 data plane using Routing Extension Header(SRH, [RFC8754]). A segment in Segment Routing is not limited to a routing/forwarding function. An SRv6 Segment can indicate functions that are executed locally in the node where they are defined.

[I-D.filsfils-spring-srv6-network-programming] describes some well-known functions and segments associated to them. SRH TLVs([RFC8754]) also provides meta-data for segment processing. All these features make SRv6 suitable to carry DetNet flows, by defining new segments associated with DetNet functions and meta data for DetNet.

This document describes the concepts needed by implementing DetNet in an SRv6-based domain and provides considerations for any corresponding implementation.

Editor's note: DetNet is limited in a controlled network domain, and it is not the only method to provide SLA guarantee. But it is a good start to discuss how to use SRv6 to have a SLA-guaranteed network service.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terminologies for DetNet go along with the definition in [RFC8655] and [RFC8754]:

DetNet domain

The portion of a network that is DetNet aware. It includes end systems and DetNet nodes

DetNet edge node

An instance of a DetNet relay node that acts as a source and/ or destination at the DetNet service sub-layer. For example, it can include a DetNet service sub-layer proxy function for DetNet service protection (e.g., the addition or removal of packet sequencing information) for one or more end systems, or starts or terminates resource allocation at the DetNet forwarding sub-layer, or aggregates DetNet services into new DetNet flows. It is analogous to a Label Edge Router (LER) or a Provider Edge (PE) router.

DetNet relay node

A DetNet node including a service sub-layer function that interconnects different DetNet forwarding sub-layer paths to provide service protection. A DetNet relay node participates in the DetNet service sub-layer. It typically incorporates DetNet forwarding sub-layer functions as well, in which case it is collocated with a transit node.

DetNet transit node

A DetNet node operating at the DetNet forwarding sub-layer, that utilizes link layer and/or network layer switching across multiple links and/or sub-networks to provide paths for DetNet service sub-layer functions. Typically provides resource allocation over those paths. An MPLS LSR is an example of a DetNet transit node.

End system

End systems of interest to this document are either sources or destinations of DetNet flows. An end system may or may not be DetNet forwarding sub-layer aware or DetNet service sub-layer aware.

DetNet service sub-layer

DetNet functionality is divided into two sub-layers. One of them is the DetNet service sub-layer, at which a DetNet service, e.g., service protection is provided.

DetNet forwarding sub-layer

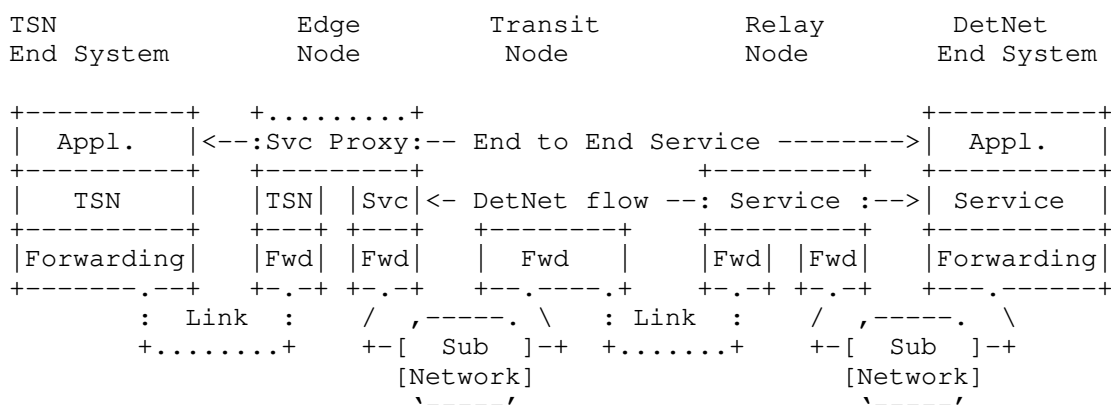
DetNet functionality is divided into two sub-layers. One of them is the DetNet forwarding sub-layer, which optionally provides resource allocation for DetNet flows over paths provided by the underlying network.

3. SRv6 for DetNet Overview

As mentioned above, there are mainly three technologies/functions defined in DetNet: Explicit Route, Resource Reservation and Service Protection. Explicit Route is the basic of the other two technologies, and guarantees the path satisfies the SLA requirement from application. Resource Reservation protects DetNet flows from network congestion, which could reduce the end-to-end latency and congestion loss; Service Protection prevents DetNet flow from random media errors and equipment failures, which makes the loss rate extremely low.

In [RFC8655], DetNet functionality is implemented in two sub-layers in the protocol stack: the DetNet service sub-layer and the DetNet forwarding sub-layer. The DetNet service sub-layer provides DetNet service, e.g., service protection. The DetNet forwarding sub-layer supports DetNet service in the underlying network, by providing explicit routes and resource allocation to DetNet flows. There is no obvious protocol stack as MPLS, in SRv6 both service sub-layer and forwarding sub-layer are implemented through SRH.

The following picture shows that a general DetNet enabled network defined in [RFC8655] :

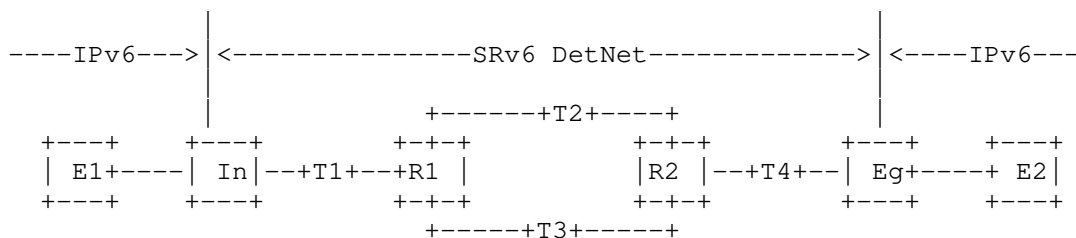


In SRv6 for DetNet, the outer IPv6 Header with the SRH is used for carrying DetNet flows. Explicit path is instantiated in the segment list of SRH, and other functions/arguments for service protection (packet replication, elimination and ordering, PREOF) and resource reservation (packet queuing and scheduling) are also defined in the specified SID. Meta data for DetNet could be instantiated in the Optional TLVs.

4. Service Protection

4.1. Service Protection Scenarios

The figure below shows that an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the outer IPv6 header+SRH of the packet is popped, and the packet is sent to the destination E2.



The DetNet packet processing is as follows:

Ingress:

Inserts the SRv6 Policy that will steer the packet from Ingress to the destination

The methods and mechanisms used for defining, instantiating and applying the policy are outside of this document. An example of policies are described in [I-D.ietf-spring-segment-routing-policy]

Flow Identification and Sequence Number are carried in the SRH.

Relay Node 1(Replication Node):

Replicates the payload and IPv6 Header with the SRH. This is a new function in the context of SRv6 Network Programming which will associate a given SID to a replication instruction in the node originating and advertising the SID. The replication instruction includes:

- * The removal of the existing IPv6+SRH header
- * The encapsulation into a new outer IPv6+SRH header. Each packet (the original and the duplicated) are encapsulated into respectively new outer IPv6+SRH headers.

Binding two different SRv6 Policies respectively to the original packet and the replicated packet, which can steer the packets from Relay Node 1 to Relay Node 2 through two tunnels.

Relay Node 2(Elimination Node):

Eliminates the redundant packets.

Binds a new SRv6 Policy to the survival packet, which steers the packet from Relay Node 2 to Egress.

Egress:

Decapsulates the outer Ipv6 header.

Sends the inter packet to the End Station 2.

The DetNet packet encapsulation is illustrated here below. It has to be noted that, in the example below, the R2 address is a SRH SID associated to a TBD function related to the packet replication the node R1 has to perform. The same (or reverse) apply to node R2 which is in charge of the discard of the duplicated packet. Here also a new function will have a new SID allocated to it and representing the delete of the duplication in R2.

End Station1 output packet: (E1,E2)

Ingress output packet: (In, T1) (R1,T1, SL=2) (E1,E2)

Transit Node1 output packet: (In, R1) (R1,T1,SL=1) (E1,E2)

Relay Node1 output packets : (R1,T2) (R2,T2,SL=2) (E1,E2),
(R1,T3) (R2,T3,SL=2) (E1,E2)

Transit Node2 output packet: (R1, R2) (R2,T2,SL=1) (E1,E2)

Transit Node3 output packet: (R1, R2) (R2,T3,SL=1) (E1,E2)

Relay Node2 output packet: (R2, T4) (Eg,T4,SL=2) (E1,E2)

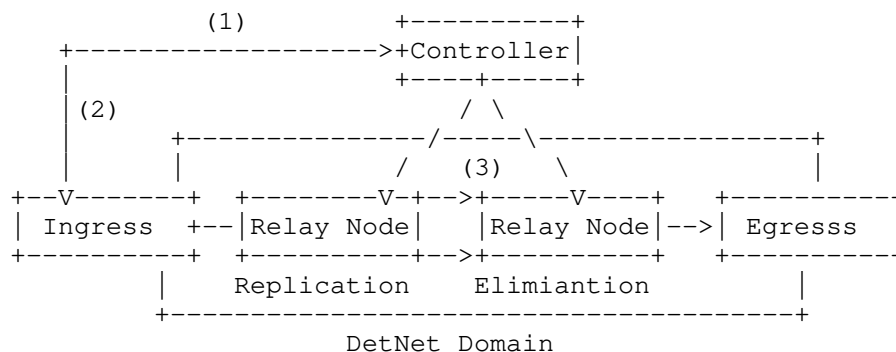
Transit Node4 output packet: (R2, Eg) (Eg,T4,SL=1) (E1,E2)

Egress out : (E1,E2)

4.2. Data Plane Considerations

Flow Identification and sequence number are necessary in the encapsulation of SRv6 for DetNet in order to support service protection. Replication nodes decide which DetNet flows are supposed to be replicated by identifying the flow with the flow identification. Elimination nodes decide whether a packet should be dropped because of redundancy by identifying the flow identification and sequence number.

4.3. Control Plane Considerations



1. Edge node->Controller: Sends a path computation requirement containing that service protection in order to have ultra-reliability through PCEP/BGP extensions.

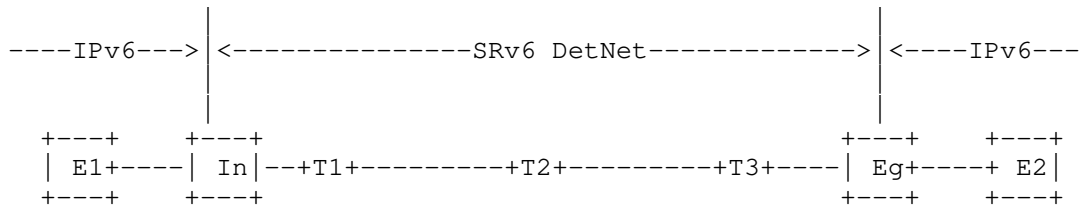
2. Controller->Edge node: Computes a P2MP2P path, including replication nodes and elimination nodes. Between a pair of replication node and elimination node, there are more than one path, and if any equipment failure happens in one of them, the DetNet service is not interrupted. Send the path computation result to the edge node through PCEP/BGP extensions.

3. Controller->Relay Node : In a P2MP2P path, every pair of replication node and elimination node should be configured to identify different DetNet flows by the different flow identifications, and the rule of sequence number should be negotiated between relay nodes. After replication or elimination, the explicit path to the next relay is also required through BGP extensions or Netconf YANG.

5. Resource Reservation

5.1. Resource Reservation Scenarios

The figure below shows that an IPv6 flow is sent out from the end station E1. The packet of the flow is encapsulated in an outer IPv6+SRH header as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the outer IPv6 header+SRH of the packet is popped, and the packet is sent to the destination E2.



The DetNet packet processing is as follows:

Ingress:

Inserts the SRv6 Policy that will steer the packet from Ingress to the destination.

The methods and mechanisms used for defining, instantiating and applying the policy are outside of this document. An example of policies are described in [I-D.ietf-spring-segment-routing-policy]

Explicit route and the corresponding resource is carried in the SRH.

T1 Node (Transit Node):

Forward the packet with the allocated resource.

This is a new function in the context of SRv6 Network Programming which will associate a given SID to a replication instruction in the node originating and advertising the SID. The instruction includes:

- * Forward the packet to the link bound to the SID
- * Use the resource when forwarding the packet

The processing of T2 and T3 Node is similar.

Egress:

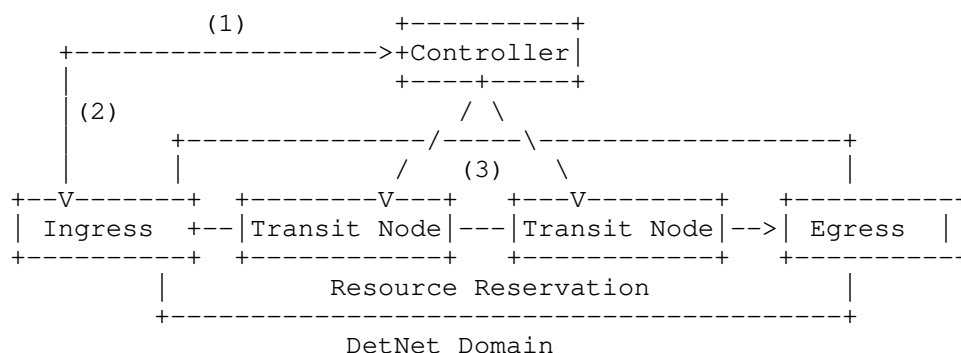
Decapsulates the outer IPv6 header.

Sends the inner packet to the End Station 2.

5.2. Data Plane Considerations

Congestion could cause uncontrolled delay and packet loss. DetNet flows are supposed to be protected from congestion, so enough resource reservation for DetNet service is necessary. Some of the resource in the network is complex and hard to quantize, e.g., packet processing resource; while some of them is easy to get and calculate, e.g., buffer size, port bandwidth and so on. In order to use the allocated resource for the DetNet flow, SRv6 for DetNet is supposed to carry the information of the resource. And the network device could transit the packet following the instruction in the SRH with the corresponding resources.

5.3. Control Plane Considerations



1. Edge node->Controller: Sends a path computation requirement containing the flow characteristics and resource reservation request through BGP/PCEP.

2. Controller->Edge node: The controller should collect the network resource information of the network, e.g., bandwidth, buffer size and so on, and maintain the resource status for every device/output port. Based on the flow characteristics, the controller should find a path which can guarantee that there are enough resources in every device/output port the flow goes through. The controller sends the path computation results back to the edge node, and update the resources status of the network through BGP/PCEP.

3. Controller->Transit Node: Every transit node along the path should be configured in order to make sure that when the DetNet flow goes through, the allocated resource for this flow is able to be used and no more resource than what is allocated will be used for the same flow through BGP/Netconf YANG.\

6. Explicit Route

SRv6 could support explicit route using segment list without extra extension. In DetNet, explicit route could be used with service protection or resource reservation. When explicit route works with service protection, a P2MP2P path is required to indicate the behavior of replication and elimination. When explicit route works with resource reservation, the explicit path indicates the nodes or links a DetNet flow goes through, and also indicates the resource allocated for the DetNet flow in the specified nodes or links.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

TBD

9. Acknowledgements

Thank you for valuable comments from James Guichard and Andrew Mails.

10. Normative References

- [I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.
- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header
(SRH)", draft-ietf-6man-segment-routing-header-26 (work in
progress), October 2019.
- [I-D.ietf-detnet-dp-sol-mpls]
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane
Encapsulation", draft-ietf-detnet-dp-sol-mpls-02 (work in
progress), March 2019.

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and
P. Mattes, "Segment Routing Policy Architecture", draft-
ietf-spring-segment-routing-policy-07 (work in progress),
May 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", RFC 8655,
DOI 10.17487/RFC8655, October 2019,
<<https://www.rfc-editor.org/info/rfc8655>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J.,
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header
(SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020,
<<https://www.rfc-editor.org/info/rfc8754>>.

Authors' Addresses

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2020

S. Hegde
S. Sangli
M. Srivastava
Juniper Networks Inc.
X. Xu
Alibaba Inc.
November 1, 2019

BGP-LS Extensions for Inter-AS TE using EPE based mechanisms
draft-hegde-idr-bgp-ls-epe-inter-as-02

Abstract

In certain network deployments, a single operator has multiple Autonomous Systems(AS) to facilitate ease of management. A multiple AS network design could also be a result of network mergers and acquisitions. In such scenarios, a centralized Inter-domain TE approach could provide most optimal allocation of resources and the most controlled path placement. BGP-LS-EPE [I-D.ietf-idr-bgp-ls-segment-routing-epe] describes an extension to BGP Link State (BGP-LS) for the advertisement of BGP Peering Segments along with their BGP peering node and inter-AS link information, so that efficient BGP Egress Peer Engineering (EPE) policies and strategies can be computed based on Segment Routing. This document describes extensions to the BGP-LS EPE to enable it to be used for inter-AS Traffic-Engineering (TE) purposes.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 4, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Reference Topology | 3 |
| 3. Fast Reroute Label | 4 |
| 4. TE Link attributes of PeerNode SID | 5 |
| 5. TE Link attributes of PeerAdj SID | 5 |
| 6. Link address TLV | 6 |
| 7. Example Advertisements | 7 |
| 8. Backward Compatibility | 9 |
| 9. Security Considerations | 9 |
| 10. IANA Considerations | 9 |
| 11. Acknowledgements | 9 |
| 12. References | 10 |
| 12.1. Normative References | 10 |
| 12.2. Informative References | 10 |
| Authors' Addresses | 10 |

1. Introduction

Segment Routing (SR) leverages source routing. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header with segment identifiers (SID). A SID can represent any instruction, topological or service-based. SR segments allows to enforce a flow through any topological path or service function while maintaining per-flow state only at the ingress node of the SR domain.

As there is no per-path state in the network, the bandwidth management for the paths is expected to be handled by a centralized entity which has a complete view of:

1. Up-to-date topology of the network
2. Resources, States and Attributes of links and nodes of the network
3. Run-time utilization/availability of resources

The BGP Link-State extensions provide mechanisms whereby link-state and TE information can be propagated in a network and a consumer of such BGP LS updates may build topology, provide bandwidth calendaring and other traffic engineering services. The centralized entity can be such a consumer (also referred to as controller). In the case of multi-AS networks, the controller needs to learn the per-AS network information and the inter-AS link information thus arriving at a consolidated Traffic Engineering Database which can be used to compute end-to-end Traffic Engineering Path. The controller can learn the topology, link-state and TE information from each of the AS networks either by participating in their IGPs or by listening to BGP LS updates [RFC7752]. Similar information about the inter-AS links can be learnt via BGP-LS EPE [I-D.ietf-idr-bgpls-segment-routing-epe] along with extensions defined in this document.

2. Reference Topology

The controller learns TE attributes of all the links, including the inter-AS links and uses the attributes to compute constrained paths. The controller should be able to correlate the inter-AS links for bidirectional connectivity from both ASes.

This document defines a new flag "F" in the Peering SIDs TLV to indicate a SID as an FRR SID. With the "F" flag set, the protection for any peering SID can be specified using another PeerAdjSID, PeerNodeSID or PeerSetSID to the controller. If the protection is achieved by fallback to local IP lookup, FRR SID SHOULD not be advertised. The link(s) represented by the FRR SID will carry the traffic when there is a failure. These SIDs are included as an FRR SIDs in the peerAdjSID, PeerNodeSID and PeerSetSID advertisements.

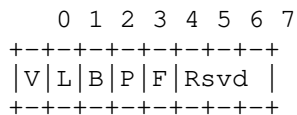


Figure 2: Peering SID TLV Flags Format

* F-Flag: FRR Label Flag: If set, the peer SID where the FRR Label appears is using backup links represented by FRR Label.

4. TE Link attributes of PeerNode SID

In any eBGP deployment, the peering session can be either single-hop or multi-hop. For single-hop eBGP sessions, the peering address is that of the directly attached interface to which the session is pinned down. For multi-hop eBGP session, the peering address is reachable over more than one interface and that the peering session is not pinned down to any of the directly attached interfaces.

A Peer Node Segment is a segment describing a peer, including the SID (PeerNodeSID) allocated to it. The link descriptors for the PeerNodeSID include the addresses used by the BGP session encoded using TLVs as defined in [RFC7752]. Since the eBGP session can be either single-hop or multi-hop, the IP address used by BGP session as local/neighbour is not sufficient to identify the underlying interface(s). Also, the controller needs to know the links associated with the PeerNodeSID, to be able derive TE link attributes. This can be achieved by including the interface local and remote addresses in the Link attributes in PeerNodeSID NLRI. This document defines a new link attribute TLV name Interface Address TLV. PeerNodeSID NLRI MAY optionally include Interface Address TLV.

5. TE Link attributes of PeerAdj SID

PeerAdjSID MUST be advertised for each inter-AS link for the purposes of inter-AS TE. The PeerAdjSID should contain link TE attributes such as bandwidth, admin-group etc. The PeerAdjSID should also

contain the local and remote interface IPv4/IPv6 addresses which is used for correlating the links. PeerNodeSID SHOULD contain the additional attribute of link local address which is used by the controller to find corresponding PeerAdjSID and hence the corresponding link TE attributes.

A peerAdj segment carries mandatory link descriptors as local and remote link id. Remote link id of the neighboring ASBR is not readily available. [I-D.ietf-idr-bgpls-segment-routing-epe] suggests to carry the value '0' for the remote link id. The Controller needs to associate the links in both directions to effectively handle failure notifications and for this purpose a unique remote link is necessary. The remote link ID cannot be manually configured on the router as the link-ids generally change over router reboot etc and hence manual configuration is operationally very difficult to manage. This document mandates advertisement of local and remote interface addresses for the inetr-AS TE purposes.

The Unnumbered interface is not in the scope of this document.

6. Link address TLV

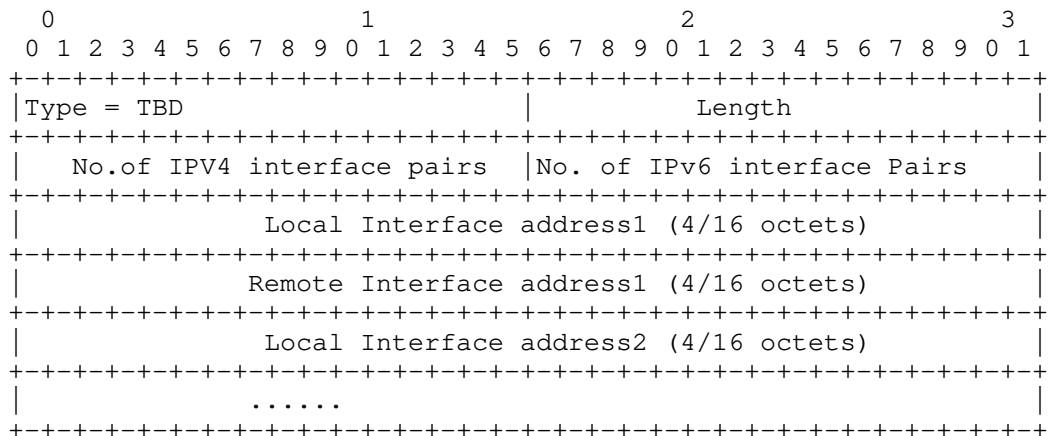


Figure 3: Link Address TLV carried as attribute

Type : TBD

Length : variable based on ipv4/ipv6 interface address

Number of IPv4 interface pairs:

Number of IPv6 interface pairs:

There may be a number of parallel interfaces and few or all of them may be used for the PeerNodeSID. These interfaces may have both IPV4 and IPV6 address or some interfaces may be IPv4 only and some IPv6 only. The total number of IPv4 and IPv6 interface address count is carried separately in above fields.

Local Interface Address :

The interface local address ipv4/ipv6 which corresponds to the PeerNodeSID MUST be specified. For IPv4, this field is 4 octets; for IPv6, this field is 16 octets.

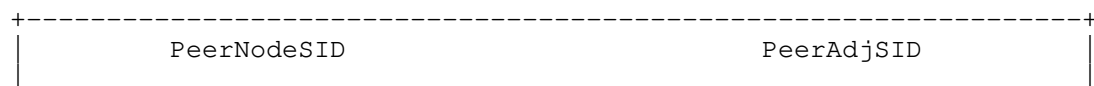
Remote Interface Address :

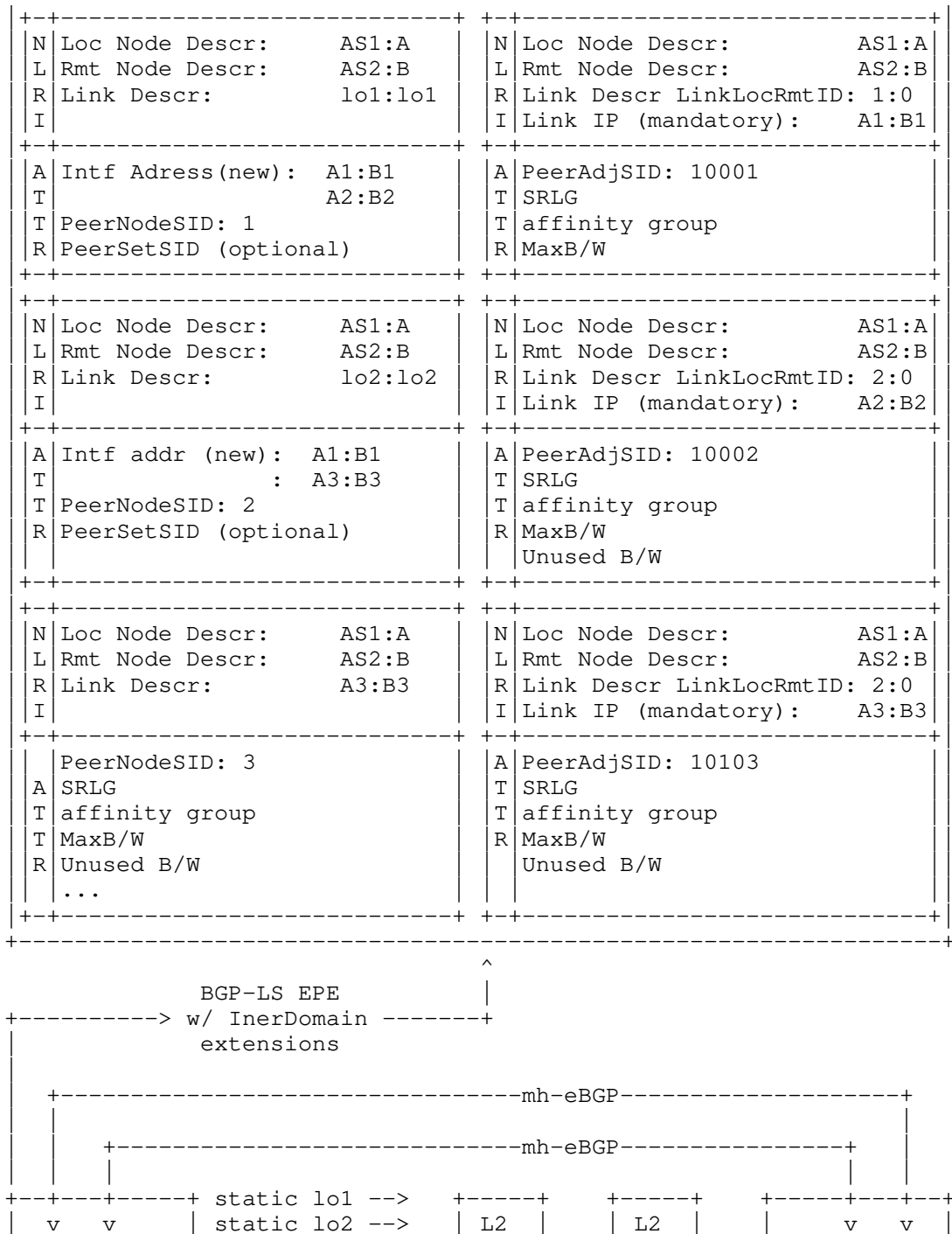
The interface remote address ipv4/ipv6 which corresponds to the PeerNodeSID MUST be specified. For IPv4, this field is 4 octets; for IPv6, this field is 16 octets.

There can be multiple Layer 3 interfaces on which a peerNodeSID loadbalances the traffic. All such interfaces local/remote address MUST be included when a Link address TLV is added. When a single Layer 3 interface consists of multiple addresses or when a link has both IPv4 and IPv6 addresses configured, It is sufficient to include one such pair (either IPV4 or IPV6) address for the PeerNodeSID advertisement. When a PeerNodeSID load-balances over few interfaces with IPv4 only address and few interfaces with IPv6 address then the Link address TLV should list all IPv4 address pairs together followed by IPv6 address pairs.

7. Example Advertisements

The below diagram represents two ASBR routers and inter-AS links between them. The inter-AS links could be connected via switches L1 and L2 as shown in the diagram or via Point-to-point links A2->B2, A3->B3 as shown in the diagram below. In the below example, let's assume peerNodeSID 1 is configured to use peerAdjSID 10002 then PeerNodeSID 1 will have the B bit set which means the PeerNodeSID 1 is eligible for backup. Label 10002 is added to the PeerNodeSID with a "F" bit set, which means 10002 is a backup for PeerNodeSID 1.





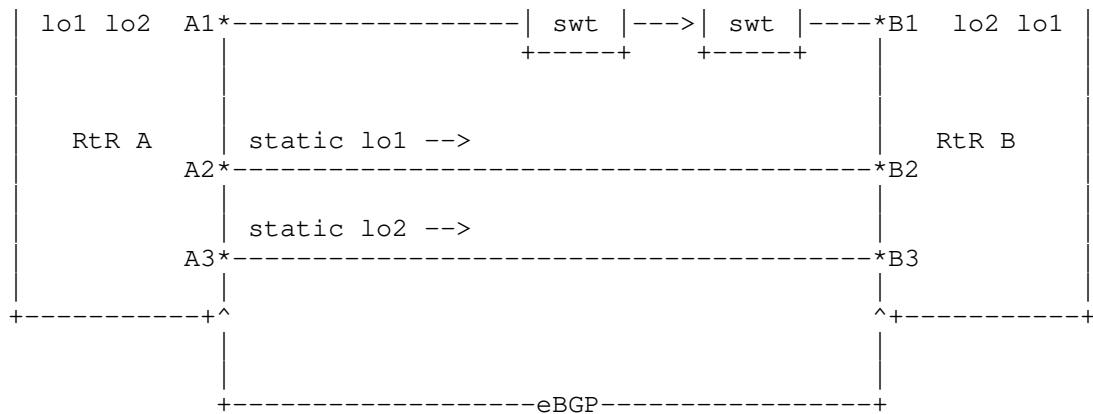


Figure 4: Example Advertisements

8. Backward Compatibility

The extension proposed in this document is backward compatible with procedures described in [I-D.ietf-idr-bgppls-segment-routing-epe] and [I-D.ietf-spring-segment-routing-central-epe]

9. Security Considerations

TBD

10. IANA Considerations

New attribute TLV in BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs registry

| TLV Code Point | Description | IS-IS TLV /Sub-TLV | Reference (RFC/Section) |
|----------------|------------------|--------------------|-------------------------|
| TBD | Link address TLV | NA | This draft |

Figure 5: IANA code point

11. Acknowledgements

Thanks to Julian Lucek and Rafal Jan Szarecki for careful review and suggestions.

12. References

12.1. Normative References

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-19 (work in progress), May 2019.

[I-D.ietf-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Dawra, G., Aries, E., and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering", draft-ietf-spring-segment-routing-central-epe-10 (work in progress), December 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.

12.2. Informative References

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Shraddha Hegde
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: shraddha@juniper.net

Srihari Sangli
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: ssangli@juniper.net

Mukul Srivastava
Juniper Networks Inc.

Email: msri@juniper.net

Xiaohu Xu
Alibaba Inc.
Beijing
China

Email: xiaohu.xxh@alibaba-inc.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: December 4, 2020

S. Hegde
S. Sangli
M. Srivastava
Juniper Networks Inc.
X. Xu
Alibaba Inc.
June 2, 2020

BGP-LS Extensions for Inter-AS TE using EPE based mechanisms
draft-hegde-idr-bgp-ls-epe-inter-as-03

Abstract

In certain network deployments, a single operator has multiple Autonomous Systems(AS) to facilitate ease of management. A multiple AS network design could also be a result of network mergers and acquisitions. In such scenarios, a centralized Inter-domain TE approach could provide most optimal allocation of resources and the most controlled path placement. BGP-LS-EPE [I-D.ietf-idr-bgp-ls-segment-routing-epe] describes an extension to BGP Link State (BGP-LS) for the advertisement of BGP Peering Segments along with their BGP peering node and inter-AS link information, so that efficient BGP Egress Peer Engineering (EPE) policies and strategies can be computed based on Segment Routing. This document describes extensions to the BGP-LS EPE to enable it to be used for inter-AS Traffic-Engineering (TE) purposes.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 4, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Reference Topology | 3 |
| 3. Fast Reroute Label | 4 |
| 4. TE Link attributes of PeerNode SID | 5 |
| 5. TE Link attributes of PeerAdj SID | 5 |
| 6. Link address TLV | 6 |
| 7. Example Advertisements | 7 |
| 8. Backward Compatibility | 9 |
| 9. Security Considerations | 9 |
| 10. IANA Considerations | 9 |
| 11. Acknowledgements | 9 |
| 12. References | 10 |
| 12.1. Normative References | 10 |
| 12.2. Informative References | 10 |
| Authors' Addresses | 10 |

1. Introduction

Segment Routing (SR) leverages source routing. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header with segment identifiers (SID). A SID can represent any instruction, topological or service-based. SR segments allows to enforce a flow through any topological path or service function while maintaining per-flow state only at the ingress node of the SR domain.

As there is no per-path state in the network, the bandwidth management for the paths is expected to be handled by a centralized entity which has a complete view of:

1. Up-to-date topology of the network
2. Resources, States and Attributes of links and nodes of the network
3. Run-time utilization/availability of resources

The BGP Link-State extensions provide mechanisms whereby link-state and TE information can be propagated in a network and a consumer of such BGP LS updates may build topology, provide bandwidth calendaring and other traffic engineering services. The centralized entity can be such a consumer (also referred to as controller). In the case of multi-AS networks, the controller needs to learn the per-AS network information and the inter-AS link information thus arriving at a consolidated Traffic Engineering Database which can be used to compute end-to-end Traffic Engineering Path. The controller can learn the topology, link-state and TE information from each of the AS networks either by participating in their IGPs or by listening to BGP LS updates [RFC7752]. Similar information about the inter-AS links can be learnt via BGP-LS EPE [I-D.ietf-idr-bgppls-segment-routing-epe] along with extensions defined in this document.

2. Reference Topology

The controller learns TE attributes of all the links, including the inter-AS links and uses the attributes to compute constrained paths. The controller should be able to correlate the inter-AS links for bidirectional connectivity from both ASes.

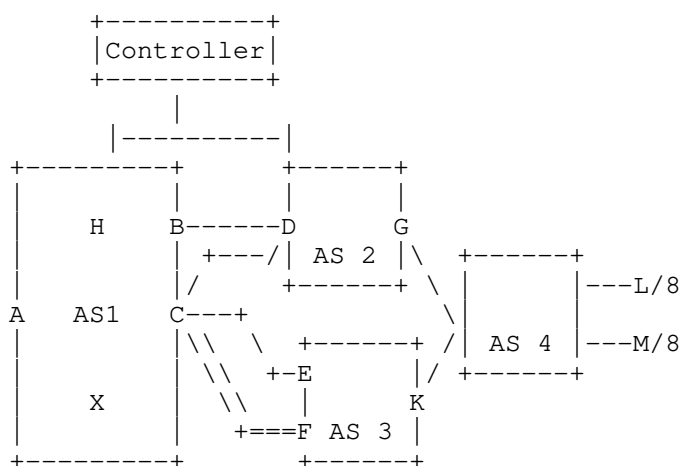


Figure 1: Reference Diagram

The reference diagram from [I-D.ietf-spring-segment-routing-central-epe] represents multiple Autonomous Systems connected to each other. When the Multiple ASes belong to same operator and are organised into separate domains for operational purposes, it is advantageous to support Traffic-Engineering across the ASes including the inter-AS links. The controller has visibility of all of the ASes by means of IGP topology exported via BGP-LS [RFC7752], or other means. In addition, the inter-AS links and the labels associated with the inter-AS links are exported via [I-D.ietf-idr-bgppls-segment-routing-epe]. The controller needs to correlate the information acquired from all of the ASes, including the inter-AS links in order to get a view of the unified topology so that it can build end-to-end Traffic-Engineered paths.

3. Fast Reroute Label

[I-D.ietf-spring-segment-routing-central-epe] section 3.6 describes mechanisms to provide Fast Reroute (FRR) protection for the EPE Labels. The BGP-LS EPE [I-D.ietf-idr-bgppls-segment-routing-epe] describes "B" bit to indicate that a PeerNodeSID or PeerAdjSID is eligible for backup. However, it does not specify what is the behaviour when the failure kicks-in. The controller needs to know which links are used for protection so that admission control and failure simulation can be done effectively and appropriate inter-AS links used for path construction.

This document defines a new flag "F" in the Peering SIDs TLV to indicate a SID as an FRR SID. With the "F" flag set, the protection for any peering SID can be specified using another PeerAdjSID, PeerNodeSID or PeerSetSID to the controller. If the protection is achieved by fallback to local IP lookup, FRR SID SHOULD not be advertised. The link(s) represented by the FRR SID will carry the traffic when there is a failure. These SIDs are included as an FRR SIDs in the peerAdjSID, PeerNodeSID and PeerSetSID advertisements.

```

      0 1 2 3 4 5 6 7
+---+---+---+---+---+---+
|V|L|B|P|F|Rsvd|
+---+---+---+---+---+---+

```

Figure 2: Peering SID TLV Flags Format

* F-Flag: FRR Label Flag: If set, the peer SID where the FRR Label appears is using backup links represented by FRR Label.

4. TE Link attributes of PeerNode SID

In any eBGP deployment, the peering session can be either single-hop or multi-hop. For single-hop eBGP sessions, the peering address is that of the directly attached interface to which the session is pinned down. For multi-hop eBGP session, the peering address is reachable over more than one interface and that the peering session is not pinned down to any of the directly attached interfaces.

A Peer Node Segment is a segment describing a peer, including the SID (PeerNodeSID) allocated to it. The link descriptors for the PeerNodeSID include the addresses used by the BGP session encoded using TLVs as defined in [RFC7752]. Since the eBGP session can be either single-hop or multi-hop, the IP address used by BGP session as local/neighbour is not sufficient to identify the underlaying interface(s). Also, the controller needs to know the links associated with the PeerNodeSID, to be able derive TE link attributes. This can be achieved by including the interface local and remote addresses in the Link attributes in PeerNodeSID NLRI. This document defines a new link attribute TLV name Interface Address TLV. PeerNodeSID NLRI MAY optionally include Interface Address TLV.

5. TE Link attributes of PeerAdj SID

PeerAdjSID MUST be advertised for each inter-AS link for the purposes of inter-AS TE. The PeerAdjSID should contain link TE attributes such as bandwidth, admin-group etc. The PeerAdjSID should also

contain the local and remote interface IPv4/IPv6 addresses which is used for correlating the links. PeerNodeSID SHOULD contain the additional attribute of link local address which is used by the controller to find corresponding PeerAdjSID and hence the corresponding link TE attributes.

A peerAdj segment carries mandatory link descriptors as local and remote link id. Remote link id of the neighboring ASBR is not readily available. [I-D.ietf-idr-bgppls-segment-routing-epe] suggests to carry the value '0' for the remote link id. The Controller needs to associate the links in both directions to effectively handle failure notifications and for this purpose a unique remote link is necessary. The remote link ID cannot be manually configured on the router as the link-ids generally change over router reboot etc and hence manual configuration is operationally very difficult to manage. This document mandates advertisement of local and remote interface addresses for the inetr-AS TE purposes.

The Unnumbered interface is not in the scope of this document.

6. Link address TLV

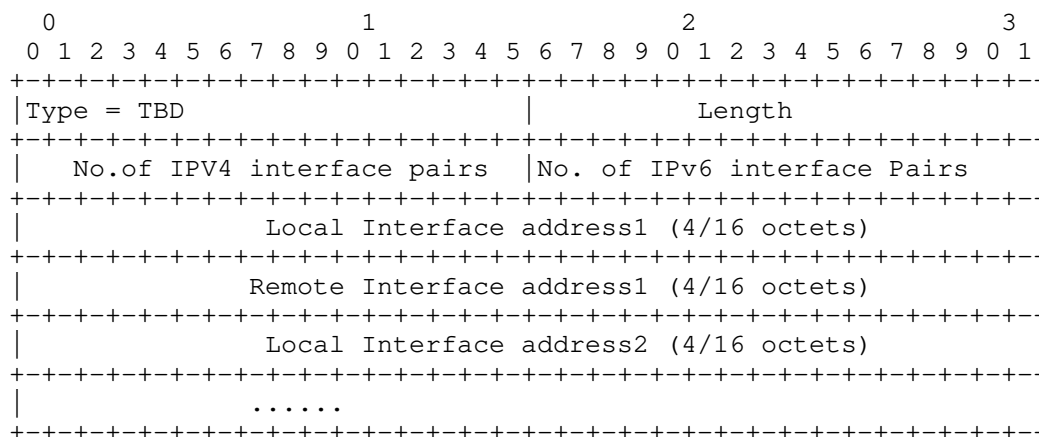


Figure 3: Link Address TLV carried as attribute

Type : TBD

Length : variable based on ipv4/ipv6 interface address

Number of IPv4 interface pairs:

Number of IPv6 interface pairs:

There may be a number of parallel interfaces and few or all of them may be used for the PeerNodeSID. These interfaces may have both IPV4 and IPV6 address or some interfaces may be IPv4 only and some IPv6 only. The total number of IPv4 and IPv6 interface address count is carried separately in above fields.

Local Interface Address :

The interface local address ipv4/ipv6 which corresponds to the PeerNodeSID MUST be specified. For IPv4, this field is 4 octets; for IPv6, this field is 16 octets.

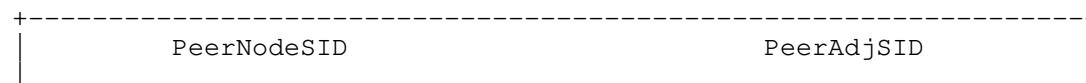
Remote Interface Address :

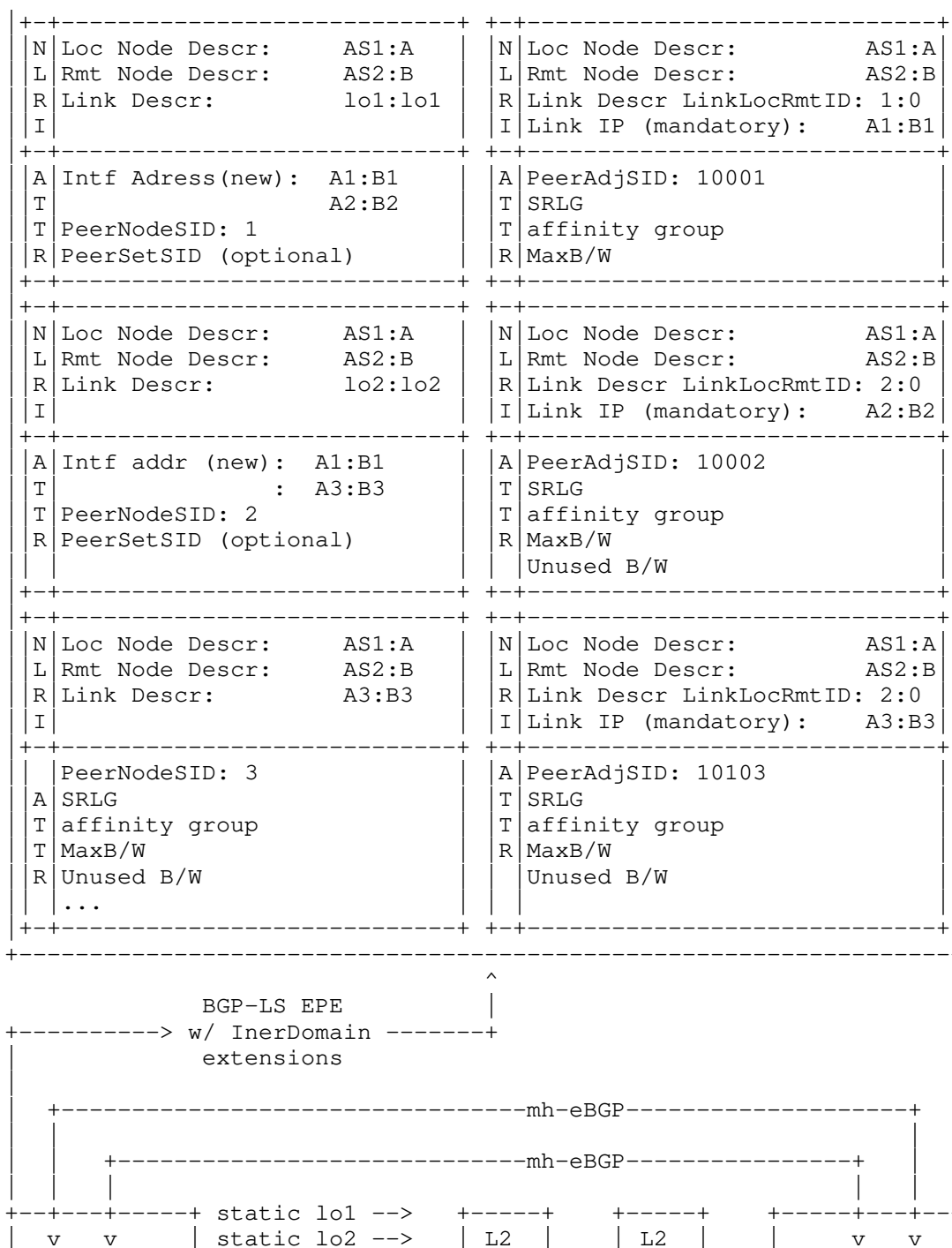
The interface remote address ipv4/ipv6 which corresponds to the PeerNodeSID MUST be specified. For IPv4, this field is 4 octets; for IPv6, this field is 16 octets.

There can be multiple Layer 3 interfaces on which a peerNodeSID loadbalances the traffic. All such interfaces local/remote address MUST be included when a Link address TLV is added. When a single Layer 3 interface consists of multiple addresses or when a link has both IPv4 and IPv6 addresses configured, It is sufficient to include one such pair (either IPV4 or IPV6) address for the PeerNodeSID advertisement. When a PeerNodeSID load-balances over few interfaces with IPv4 only address and few interfaces with IPv6 address then the Link address TLV should list all IPv4 address pairs together followed by IPv6 address pairs.

7. Example Advertisements

The below diagram represents two ASBR routers and inter-AS links between them. The inter-AS links could be connected via switches L1 and L2 as shown in the diagram or via Point-to-point links A2->B2, A3->B3 as shown in the diagram below. In the below example, lets assume peerNodeSID 1 is configured to use peerAdjSID 10002 then PeerNodeSID 1 will have the B bit set which means the PeerNodeSID 1 is eligible for backup. Label 10002 is added to the PeerNodeSID with a "F" bit set, which means 10002 is a backup for PeerNodeSID 1.





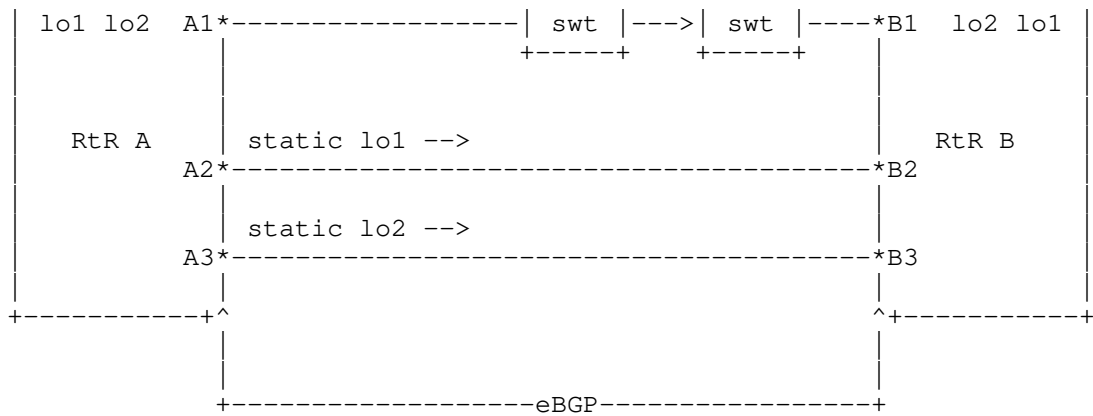


Figure 4: Example Advertisements

8. Backward Compatibility

The extension proposed in this document is backward compatible with procedures described in [I-D.ietf-idr-bgppls-segment-routing-epe] and [I-D.ietf-spring-segment-routing-central-epe]

9. Security Considerations

TBD

10. IANA Considerations

New attribute TLV in BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs registry

| TLV Code Point | Description | IS-IS TLV /Sub-TLV | Reference (RFC/Section) |
|----------------|------------------|--------------------|-------------------------|
| TBD | Link address TLV | NA | This draft |

Figure 5: IANA code point

11. Acknowledgements

Thanks to Julian Lucek and Rafal Jan Szarecki for careful review and suggestions.

12. References

12.1. Normative References

- [I-D.ietf-idr-bgppls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgppls-segment-routing-epe-19 (work in progress), May 2019.
- [I-D.ietf-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Dawra, G., Aries, E., and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering", draft-ietf-spring-segment-routing-central-epe-10 (work in progress), December 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.

12.2. Informative References

- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Shraddha Hegde
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: shraddha@juniper.net

Srihari Sangli
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: ssangli@juniper.net

Mukul Srivastava
Juniper Networks Inc.

Email: msri@juniper.net

Xiaohu Xu
Alibaba Inc.
Beijing
China

Email: xiaohu.xxh@alibaba-inc.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2020

C. Filsfils, Ed.
P. Camarillo, Ed.
Cisco Systems, Inc.
J. Leddy
Individual Contributor
D. Voyer
Bell Canada
S. Matsushima
SoftBank
Z. Li
Huawei Technologies
October 24, 2019

SRv6 Network Programming
draft-ietf-spring-srv6-network-programming-05

Abstract

This document describes the SRv6 network programming concept and its most basic functions.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 4 |
| 2. Terminology | 4 |
| 3. SRv6 SID | 6 |
| 3.1. SID format | 6 |
| 3.2. SID reachability | 7 |
| 4. Behaviors associated with a SID | 8 |
| 4.1. End: Endpoint | 9 |
| 4.2. End.X: Layer-3 cross-connect | 10 |
| 4.3. End.T: Specific IPv6 table lookup | 11 |
| 4.4. End.DX6: Decapsulation and IPv6 cross-connect | 12 |
| 4.5. End.DX4: Decapsulation and IPv4 cross-connect | 13 |
| 4.6. End.DT6: Decapsulation and specific IPv6 table lookup | 14 |
| 4.7. End.DT4: Decapsulation and specific IPv4 table lookup | 15 |
| 4.8. End.DT46: Decapsulation and specific IP table lookup | 16 |
| 4.9. End.DX2: Decapsulation and L2 cross-connect | 17 |
| 4.10. End.DX2V: Decapsulation and VLAN L2 table lookup | 18 |
| 4.11. End.DT2U: Decapsulation and unicast MAC L2 table lookup | 18 |
| 4.12. End.DT2M: Decapsulation and L2 table flooding | 19 |
| 4.13. End.B6.Encaps: Endpoint bound to an SRv6 policy w/ encaps | 20 |
| 4.14. End.B6.Encaps.Red: [...] with reduced SRH | 21 |
| 4.15. End.BM: Endpoint bound to an SR-MPLS policy | 22 |
| 4.16. Flavors | 24 |
| 4.16.1. PSP: Penultimate Segment Pop of the SRH | 24 |
| 4.16.2. USP: Ultimate Segment Pop of the SRH | 24 |
| 4.16.3. USD: Ultimate Segment Decapsulation | 24 |
| 5. Transit behaviors | 26 |
| 5.1. T: Transit behavior | 26 |
| 5.2. T.Encaps: Transit with encapsulation in an SRv6 Policy | 26 |
| 5.3. T.Encaps.Red: Transit with reduced encapsulation | 27 |
| 5.4. T.Encaps.L2: Transit with encapsulation of L2 frames | 28 |
| 5.5. T.Encaps.L2.Red: Transit with reduced encaps of L2 frames | 28 |

| | | |
|-------|--|----|
| 6. | Operation | 29 |
| 6.1. | Counters | 29 |
| 6.2. | Flow-based hash computation | 29 |
| 6.3. | OAM | 29 |
| 7. | Basic security for intra-domain deployment | 30 |
| 8. | Control Plane | 30 |
| 8.1. | IGP | 30 |
| 8.2. | BGP-LS | 31 |
| 8.3. | BGP IP/VPN/EVPN | 31 |
| 8.4. | Summary | 31 |
| 9. | IANA Considerations | 32 |
| 10. | Acknowledgements | 35 |
| 11. | Contributors | 35 |
| 12. | References | 38 |
| 12.1. | Normative References | 38 |
| 12.2. | Informative References | 38 |
| | Authors' Addresses | 40 |

1. Introduction

Segment Routing leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the segment list to any complex user-defined behavior. The network programming consists in combining segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document defines the SRv6 Network Programming concept and aims at standardizing the main segment routing behaviors to enable the creation of interoperable overlays with underlay optimization and service programming.

The companion document

[I-D.filsfils-spring-srv6-net-pgm-illustration] illustrates the concepts defined in this document.

Familiarity with the Segment Routing Header

[I-D.ietf-6man-segment-routing-header] is assumed.

2. Terminology

Terminology used within this document is defined in detail in [RFC8402]. Specifically, the terms: Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy.

SRH: Segment Routing Header as defined in

[I-D.ietf-6man-segment-routing-header]. We assume that the SRH may be present multiple times inside each packet.

NH: Next-header field of the IPv6 header. NH=SRH means that the next-header of the IPv6 header is Routing Header for IPv6(43) with the Type field set to 4.

SL: The Segments Left field of the SRH

FIB: Forwarding Information Base. A FIB lookup is a lookup in the forwarding table.

SA: Source Address

DA: Destination Address

SRv6 SID function: The function part of the SID is an opaque identification of a local behavior bound to the SID. It is formally defined in Section 3.1 of this document.

SRv6 segment behavior: A packet processing behavior executed at an SRv6 segment endpoint. Section 4 of this document defines behaviors related to traffic-engineering and both L2VPN and L3VPN use-cases. Other behaviors (e.g. service programming) are outside the scope of this document.

An SR Policy is resolved to a SID list. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Address is DA, and next-header is SRH
- SRH with SID list <S1, S2, S3> with Segments Left = SL
- Note the difference between the <> and () symbols: <S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID to traverse. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of the detailed packet behavior, the (S3, S2, S1; SL) notation is more convenient.
- The payload of the packet is omitted.

When a packet is intercepted on a wire, it is possible that SRH[SL] is different from the DA.

3. SRv6 SID

As introduced in RFC8402 an SRv6 Segment Identifier is a 128-bit value.

When an SRv6 SID is in the Destination Address field of an IPv6 header of a packet, it is routed through an IPv6 network as an IPv6 address.

Its processing is defined in [I-D.ietf-6man-segment-routing-header] section 4.3 and reproduced here as a reminder.

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

- A FIB entry that represents a locally instantiated SRv6 SID
- A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- A FIB entry that represents a non-local route
- No Match

This document formally defines behaviors and parameters for SRv6 SIDs.

3.1. SID format

An SRv6 SID is represented as LOC:FUNCT where LOC (locator) is the L most significant bits and FUNCT (function) is the 128-L least significant bits of the SID. L is called the locator length and is flexible. Each operator is free to use the locator length it chooses. Most often the locator is routable and leads to the node which instantiates that SID. A control-plane protocol might represent the locator as B:N where B is the SRv6 SID block (IPv6 subnet allocated for SRv6 SIDs by the operator) and N is the identifier of the parent node.

The function part of the SID is an opaque identification of a local behavior bound to the SID. The FUNCT value zero is invalid.

The terminology "function" refers to the bit-string in the SRv6 SID. The terminology "behavior" identifies the pseudocode bound to the SID. The behaviors are defined in Section 4 of this document.

A behavior may require additional arguments that would be placed immediately after the FUNCT. In such case, the SRv6 SID will have the form LOC:FUNCT:ARGS::. For this reason, the SRv6 SIDs are matched on a per longest-prefix-match basis.

ARG may contain information related to the flow, service, or any other information required by FUNCT. The ARG value of a routed SID SHOULD remain constant among packets in a given flow. Varying ARG values among packets in a flow may result in different ECMP hashing and cause re-ordering.

3.2. SID reachability

Most often, the node N would advertise IPv6 prefix(es) matching the LOC parts covering its SIDs or shorter-mask prefix. The distribution of these advertisements and calculation of their reachability are routing protocol specific aspects that are outside the scope of this document.

An SRv6 SID is said to be routed if its SID belongs to an IPv6 prefix advertised via a routing protocol. An SRv6 SID that does not fulfill this condition is non-routed.

Let's provide a classic illustration:

Node N is configured explicitly with two SIDs: 2001:DB8:B:1:100:: and 2001:DB8:B:2:101::.

The network learns about a path to 2001:DB8:B:1::/64 via the IGP and hence a packet destined to 2001:DB8:B:1:100:: would be routed up to N. The network does not learn about a path to 2001:DB8:B:2::/64 via the IGP and hence a packet destined to 2001:DB8:B:2:101:: would not be routed up to N.

A packet could be steered to a non-routed SID 2001:DB8:B:2:101:: by using a SID list <...,2001:DB8:B:1:100::,2001:DB8:B:2:101::,...> where the non-routed SID is preceded by a routed SID to the same node. Routed and non-routed SRv6 SIDs are the SRv6 instantiation of global and local segments, respectively [RFC8402].

4. Behaviors associated with a SID

Each FIB entry indicates the behavior associated with a SID instance and its parameters.

We define hereafter a set of well-known behaviors that can be associated with a SID.

| | |
|-------------------|---|
| End | Endpoint function |
| | The SRv6 instantiation of a prefix SID |
| End.X | Endpoint with Layer-3 cross-connect |
| | The SRv6 instantiation of a Adj SID |
| End.T | Endpoint with specific IPv6 table lookup |
| End.DX6 | Endpoint with decaps and IPv6 cross-connect |
| | e.g. IPv6-L3VPN (equivalent to per-CE VPN label) |
| End.DX4 | Endpoint with decaps and IPv4 cross-connect |
| | e.g. IPv4-L3VPN (equivalent to per-CE VPN label) |
| End.DT6 | Endpoint with decaps and IPv6 table lookup |
| | e.g. IPv6-L3VPN (equivalent to per-VRF VPN label) |
| End.DT4 | Endpoint with decaps and IPv4 table lookup |
| | e.g. IPv4-L3VPN (equivalent to per-VRF VPN label) |
| End.DT46 | Endpoint with decaps and IP table lookup |
| | e.g. IP-L3VPN (equivalent to per-VRF VPN label) |
| End.DX2 | Endpoint with decaps and L2 cross-connect |
| | e.g. L2VPN use-case |
| End.DX2V | Endpoint with decaps and VLAN L2 table lookup |
| | e.g. EVPN Flexible cross-connect use-case |
| End.DT2U | Endpoint with decaps and unicast MAC L2table lookup |
| | e.g. EVPN Bridging unicast use-case |
| End.DT2M | Endpoint with decaps and L2 table flooding |
| | e.g. EVPN Bridging BUM use-case with ESI filtering |
| End.B6.Encaps | Endpoint bound to an SRv6 policy with encaps |
| | SRv6 instantiation of a Binding SID |
| End.B6.Encaps.RED | [...] with reduced SRH insertion |
| | SRv6 instantiation of a Binding SID |
| End.BM | Endpoint bound to an SR-MPLS Policy |
| | SRv6 instantiation of an SR-MPLS Binding SID |

The list is not exhaustive. In practice, any function can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex processing on the packet.

The following subsections detail the behavior that a node (N) binds to a SID (S).

At the end of this section, we also present some flavors of these well-known behaviors.

4.1. End: Endpoint

The Endpoint behavior ("End" for short) is the most basic behavior. It is the instantiation of a Prefix-SID [RFC8402].

It does not allow for decapsulation of an outer header nor the removal of an SRH. As a consequence, an End SID cannot be the last SID of a SID list and cannot be the DA of a packet without an SRH (unless combined with the PSP, USP or USD flavors Section 4.16).

The following defines SRH processing and, if SRH is not present, upper-layer header processing when a matched FIB entry represents a locally instantiated End SID.

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet
S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Resubmit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S16. }
```

Notes:

The End behavior operates on the same FIB table (i.e. VRF, L3 relay id) associated to the packet. Hence the FIB lookup on line S15 is done in the same FIB table as the ingress interface.

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code TBD-SRH (SR Upper-layer Header Error) and Pointer set to the offset of the upper-layer header.

4.2. End.X: Layer-3 cross-connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" behavior (End.X for short) is a variant of the End behavior.

It is the SRv6 instantiation of an Adjacency-SID [RFC8402] and it is required to express any traffic-engineering policy.

An instance of the End.X behavior is associated with a set of J of one or more Layer-3 adjacencies.

When N receives a packet destined to S and S is a local End.X SID, the line S15 from the End processing is replaced by the following:

S15. Set the packet's egress adjacency to J

Notes:

S15. If the set J contains several L3 adjacencies, then one element of the set is selected based on a hash of the packet's header
Section 6.2.

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.X SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code "SR Upper-layer Header Error", Pointer set to the offset of the upper-layer header.

Note that the End.X SID cannot be the last SID of a SID list and cannot be the DA of a packet without an SRH (unless combined with the PSP, USP or USD flavors Section 4.16). Hence the Upper-layer header should never be processed.

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N may allocate up to 11 End.X local SIDs: one for the bundle(LAG) itself and then up to one for each Layer-2 member link.

The End.X behavior can be also associated with a BGP Next-Hop, in which case it is the SRv6 instantiation of the BGP Peering Segments [RFC8402].

4.3. End.T: Specific IPv6 table lookup

The "Endpoint with specific IPv6 table lookup" behavior (End.T for short) is a variant of the End behavior.

The End.T behavior is used for multi-table operation in the core. For this reason, an instance of the End.T behavior must be associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.T SID, the line S15 from the End processing is replaced by the following:

- S15.1. Set the packet's associated FIB table to T
- S15.2. Resubmit the packet to the egress IPv6 FIB lookup and transmission to the new destination

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.T SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code "SR Upper-layer Header Error", Pointer set to the offset of the upper-layer header.

Note that the End.T SID cannot be the last SID of a SID list and cannot be the DA of a packet without an SRH (unless combined with the PSP, USP or USD flavors Section 4.16). Hence the Upper-layer header should never be processed.

4.4. End.DX6: Decapsulation and IPv6 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" behavior (End.DX6 for short) is a variant of the End.X behavior.

One of the applications of the End.DX6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX6 SID must be the last segment in a SR Policy, and it must be associated with one or more L3 IPv6 adjacencies J.

When N receives a packet destined to S and S is a local End.DX6 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX6 SID, the following must be done.

```
S01. If (Upper-Layer Header type != 41) {
S02.   Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Forward the exposed IPv6 packet to the L3 adjacency J
```

Notes:

S01. 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

S05. If the End.DX6 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header Section 6.2.

4.5. End.DX4: Decapsulation and IPv4 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" behavior (End.DX4 for short) is a variant of the End.X behavior.

One of the applications of the End.DX4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX4 SID must be the last segment in a SR Policy, and it must be associated with one or more L3 IPv4 adjacencies J.

When N receives a packet destined to S and S is a local End.DX4 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX4 SID, the following must be done.

```
S01. If (Upper-Layer Header type != 4) {
S02.   Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Forward the exposed IPv4 packet to the L3 adjacency J
```

Notes:

S01. 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

S05. If the End.DX4 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header Section 6.2.

4.6. End.DT6: Decapsulation and specific IPv6 table lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" behavior (End.DT6 for short) is a variant of the End.T behavior.

One of the applications of the End.DT6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case and End.DT6 supports the equivalent of an IPv6inIPv6 decapsulation (without VPN/tenant implication).

The End.DT6 SID must be the last segment in a SR Policy, and a SID instance must be associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.DT6 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT6 SID, N does the following:

```
S01. If (Upper-Layer Header type != 41) {
S02.   Send an ICMP Parameter Problem message to the Source Address
           Code TBD-SRH (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header,
           interrupt packet processing and discard the packet
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Set the packet's associated FIB table to T
S06. Resubmit the packet to the egress IPv6 FIB lookup and
      transmission to the new destination
```

4.7. End.DT4: Decapsulation and specific IPv4 table lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" behavior (End.DT4 for short) is a variant of the End behavior.

One of the applications of the End.DT4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case an End.DT4 supports the equivalent of an IPv4inIPv6 decapsulation (without VPN/tenant implication).

The End.DT4 SID must be the last segment in a SR Policy, and a SID instance must be associated with an IPv4 FIB table T.

When N receives a packet destined to S and S is a local End.DT4 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT4 SID, N does the following:

```
S01. If (Upper-Layer Header type != 4) {
S02.   Send an ICMP Parameter Problem message to the Source Address
           Code TBD-SRH (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header,
           interrupt packet processing and discard the packet
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Set the packet's associated FIB table to T
S06. Resubmit the packet to the egress IPv4 FIB lookup and
      transmission to the new destination
```

4.8. End.DT46: Decapsulation and specific IP table lookup

The "Endpoint with decapsulation and specific IP table lookup" behavior (End.DT46 for short) is a variant of the End.DT4 and End.DT6 behavior.

One of the applications of the End.DT46 behavior is the L3VPN use-case where a FIB lookup in a specific IP tenant table at the egress PE is required. This would be equivalent to single per-VRF VPN label (for IPv4 and IPv6) in MPLS[RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case an End.DT46 supports the equivalent of an IPinIPv6 decapsulation(without VPN/tenant implication).

The End.DT46 SID must be the last segment in a SR Policy, and a SID instance must be associated with an IPv4 FIB table T4 and an IPv6 FIB table T6.

When N receives a packet destined to S and S is a local End.DT46 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT46 SID, N does the following:


```
S01. If (Upper-layer Header type == 4) {
S02.     Remove the outer IPv6 Header with all its extension headers
S03.     Set the packet's associated FIB table to T4
S04.     Resubmit the packet to the egress IPv4 FIB lookup and
           transmission to the new destination
S05. } Else if (Upper-layer Header type == 41) {
S06.     Remove the outer IPv6 Header with all its extension headers
S07.     Set the packet's associated FIB table to T6
S08.     Resubmit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
S09. } Else {
S10.     Send an ICMP Parameter Problem message to the Source Address
           Code TBD-SRH (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header,
           interrupt packet processing and discard the packet
S11. }
```

4.9. End.DX2: Decapsulation and L2 cross-connect

The "Endpoint with decapsulation and Layer-2 cross-connect to an outgoing L2 interface (OIF)" (End.DX2 for short) is a variant of the endpoint behavior.

One of the applications of the End.DX2 behavior is the L2VPN/EVPN VPWS use-case.

The End.DX2 SID must be the last segment in a SR Policy, and it must be associated with one outgoing interface I.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
S01. When an SRH is processed {
S02.     If (Segments Left != 0) {
S03.         Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet
S04.     }
S05.     Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX2 SID, the following must be done.

```
S01. If (Upper-Layer Header type != TBD1) {
S02.     Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S03. }
S04. Remove the outer IPv6 Header with all its extension headers and
    forward the ethernet frame to the OIF I.
```

Notes:

S04. An End.DX2 behavior could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

4.10. End.DX2V: Decapsulation and VLAN L2 table lookup

The "Endpoint with decapsulation and specific VLAN table lookup" behavior (End.DX2V for short) is a variant of the End.DX2 behavior.

One of the applications of the End.DX2V behavior is the EVPN Flexible cross-connect use-case. The End.DX2V behavior is used to perform a lookup of the ethernet frame VLANs in a particular L2 table. Any SID instance of the End.DX2V behavior must be associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DX2 SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is modified as follows:

```
S04. Remove the outer IPv6 Header with all its extension headers,
    lookup the exposed inner VLANs in L2 table T, and forward
    via the matched table entry.
```

Notes:

An End.DX2V behavior could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

4.11. End.DT2U: Decapsulation and unicast MAC L2 table lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" behavior (End.DT2U for short) is a variant of the End behavior.

One of the applications of the End.DT2U behavior is the EVPN Bridging unicast . Any SID instance of the End.DT2U behavior must be associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2U SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type != TBD1) {
S02.   Send an ICMP Parameter Problem message to the Source Address
       Code TBD-SRH (SR Upper-layer Header Error),
       Pointer set to the offset of the upper-layer header,
       interrupt packet processing and discard the packet
S03. }
S04. Remove the IPv6 header and all its extension headers
S05. Learn the exposed inner MAC Source Address in L2 Table T
S06. Lookup the exposed inner MAC Destination Address in L2 Table T
S07. If (matched entry in T) {
S08.   Forward via the matched table T entry
S09. } Else {
S10.   Forward via all L2OIFs entries in table T
S11. }
```

Notes:

S05. In EVPN, the learning of the exposed inner MAC SA is done via control plane.

4.12. End.DT2M: Decapsulation and L2 table flooding

The "Endpoint with decapsulation and specific L2 table flooding" behavior (End.DT2M for short) is a variant of the End.DT2U behavior.

Two of the applications of the End.DT2M behavior are the EVPN Bridging BUM with ESI filtering and the EVPN ETREE use-cases.

Any SID instance of this behavior must be associated with a L2 table T. Additionally the behavior may take an argument: "Arg.FE2". It is an argument specific to EVPN ESI filtering and EVPN-ETREE used to exclude specific OIF (or set of OIFs) from L2 table T flooding.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2M SID, the processing is identical to the End.DT2M behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type != TBD1) {  
S02.     Send an ICMP Parameter Problem message to the Source Address  
        Code TBD-SRH (SR Upper-layer Header Error),  
        Pointer set to the offset of the upper-layer header,  
        interrupt packet processing and discard the packet  
S03. }  
S04. Remove the IPv6 header and all its extension headers  
S05. Learn the exposed inner MAC Source Address in L2 Table T  
S06. Forward via all L2 OIFs excluding the one specified in Arg.F2
```

Notes:

S05. In EVPN, the learning of the exposed inner MAC SA is done via control plane

4.13. End.B6.Encaps: Endpoint bound to an SRv6 policy w/ encaps

This is a variation of the End behavior.

One of its applications is to express scalable traffic-engineering policies across multiple domains. It is the one of the SRv6 instantiations of a Binding SID [RFC8402].

An End.B6.Encaps SID is never the last segment in a SID list. Any SID instantiation must be associated with an SR Policy B[I-D.ietf-spring-segment-routing-policy] and a source address A.

When N receives a packet whose IPv6 DA is S and S is a local End.B6.Encaps SID, does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet
S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Push a new IPv6 header with its own SRH containing B
S15.   Set the outer IPv6 SA to A
S16.   Set the outer IPv6 DA to the first SID of B
S17.   Set the outer PayloadLength, Traffic Class, FlowLabel and
        Next-Header fields
S18.   Resubmit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S19. }
```

Notes:

S13. The SRH MAY be omitted when the SRv6 Policy B only contains one SID and there is no need to use any flag, tag or TLV.
S16. The Payload Length, Traffic Class and Next-Header fields are set as per [RFC2473]. The Flow Label is computed as per [RFC6437].

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.B6.Encaps SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code "SR Upper-layer Header Error", Pointer set to the offset of the upper-layer header.

4.14. End.B6.Encaps.Red: [...] with reduced SRH

This is an optimization of the End.B6.Encaps behavior.

End.B6.Encaps.Red reduces the size of the SRH by one SID by avoiding the insertion of the first SID in the outer SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

The new SRH is created as described in Section 4.1.1 of [I-D.ietf-6man-segment-routing-header].

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

4.15. End.BM: Endpoint bound to an SR-MPLS policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End behavior.

The End.BM behavior is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing. This is an SRv6 instantiation of an SR-MPLS Binding SID [RFC8402].

An End.BM SID is never the last SID, and any SID instantiation must be associated with an SR-MPLS Policy
B[I-D.ietf-spring-segment-routing-policy].

When N receives a packet whose IPv6 DA is S and S is a local End.BM SID, does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Send an ICMP Parameter Problem message to the Source Address
        Code TBD-SRH (SR Upper-layer Header Error),
        Pointer set to the offset of the upper-layer header,
        interrupt packet processing and discard the packet
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet
S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Push a the MPLS label stack for B
S15.   Submit the packet to the MPLS engine for transmission to the
        topmost label.
S16. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.BM SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code "SR Upper-layer Header Error", Pointer set to the offset of the upper-layer header.

4.16. Flavors

The PSP, USP and USD flavors are variants of the End, End.X and End.T behaviors. For each of these behaviors these flavors may be supported for a SID either individually or in combinations.

4.16.1. PSP: Penultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: after the instruction "S14. Update IPv6 DA with Segment List[Segments Left]" is executed, the following instructions must be executed as well:

```
S14.1.  If (updated SL == 0) {
S14.2.      Pop the SRH
S14.3.  }
```

4.16.2. USP: Ultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
S02.  If (Segments Left == 0) {
S03.      Pop the SRH
S04.  }
```

4.16.3. USD: Ultimate Segment Decapsulation

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
S02.  If (Segments Left == 0) {
S03.      Skip the SRH processing and proceed to the next header
S04.  }
```

Further on, the Upper-layer header processing of the End, End.X and End.T behaviors are modified as follows:


```
End:
S01. If (Upper-layer Header type == 41 || 4) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Resubmit the packet to the egress IP FIB lookup and
       transmission to the new destination
S04. } Else {
S05.   Send an ICMP Parameter Problem message to the Source Address
       Code TBD-SRH (SR Upper-layer Header Error),
       Pointer set to the offset of the upper-layer header,
       interrupt packet processing and discard the packet
S06. }
```

```
End.T:
S01. If (Upper-layer Header type == 41 || 4) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Resubmit the packet to the egress IP FIB lookup and
       Transmission to the new destination
S05. } Else {
S06.   Send an ICMP Parameter Problem message to the Source Address
       Code TBD-SRH (SR Upper-layer Header Error),
       Pointer set to the offset of the upper-layer header,
       interrupt packet processing and discard the packet
S07. }
```

```
End.X:
S01. If (Upper-layer Header type == 41 || 4) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IP packet to the L3 adjacency J
S04. } Else {
S05.   Send an ICMP Parameter Problem message to the Source Address
       Code TBD-SRH (SR Upper-layer Header Error),
       Pointer set to the offset of the upper-layer header,
       interrupt packet processing and discard the packet
S06. }
```

5. Transit behaviors

We define hereafter the set of basic transit behaviors. These behaviors are not bound to a SID and they correspond to source SR nodes or transit nodes [I-D.ietf-6man-segment-routing-header].

| | |
|-----------------|--|
| T | Transit behavior |
| T.Encaps | Transit behavior with encapsulation in an SRv6 policy |
| T.Encaps.Red | Transit behavior with reduced encaps in an SRv6 policy |
| T.Encaps.L2 | T.Encaps applied to received L2 frames |
| T.Encaps.L2.Red | T.Encaps.Red applied to received L2 frames |

This list can be expanded in case any new functionality requires it.

5.1. T: Transit behavior

As per [RFC8200], if a node N receives a packet (A, S2) (S3, S2, S1; SL=1) and S2 is neither a local address nor a local SID of N then N forwards the packet without inspecting the SRH.

This means that N treats the following two packets P1 and P2 with the same performance:

P1 = (A, S2)

P2 = (A, S2) (S3, S2, S1; SL=1)

A transit node does not need to count by default the amount of transit traffic with an SRH extension header. This accounting might be enabled as an optional behavior.

A transit node MUST include the outer flow label in its ECMP load-balancing hash [RFC6437].

5.2. T.Encaps: Transit with encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2) (B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SR Encapsulation Policy with a Source Address T and a Segment list <S1, S2, S3>.

The T.Encaps transit encapsulation behavior is defined as follows:

1. push an IPv6 header with its own SRH (S3, S2, S1; SL=2)
2. set outer IPv6 SA = T and outer IPv6 DA = S1
3. set outer payload length, traffic class and flow label ;; Ref1,2
4. update the Next-Header value ;; Ref1
5. decrement inner Hop Limit or TTL ;; Ref1
6. forward along the shortest path to S1

After the T.Encaps behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)
- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The T.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployments.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

5.3. T.Encaps.Red: Transit with reduced encapsulation

The T.Encaps.Red behavior is an optimization of the T.Encaps behavior. It is defined as follows:

1. push an IPv6 header with its own SRH (S3, S2; SL=2)
2. set outer IPv6 SA = T and outer IPv6 DA = S1
3. set outer payload length, traffic class and flow label ;; Ref1,2
4. update the Next-Header value ;; Ref1
5. decrement inner Hop Limit or TTL ;; Ref1
6. forward along the shortest path to S1

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

T.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

After the T.Encaps.Red behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)
- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.4. T.Encaps.L2: Transit with encapsulation of L2 frames

While T.Encaps encapsulates the received IP packet, T.Encaps.L2 encapsulates the received L2 frame (i.e. the received ethernet header and its optional VLAN header is in the payload of the outer packet).

If the outer header is pushed without SRH, then the DA must be a SID of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header must be TBD1. The received Ethernet frame follows the IPv6 header and its extension headers.

Else, if the outer header is pushed with an SRH, then the last SID of the SRH must be of type End.DX2, End.DX2V, End.DT2U or End.DT2M and the next-header of the SRH must be TBD1. The received Ethernet frame follows the IPv6 header and its extension headers.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

The preamble and frame check sequence (FCS) are stripped from the Ethernet frame upon encapsulation. The egress behavior End.DX2, End.DX2V, End.DT2U or End.DT2M regenerates the preamble or FCS before forwarding the frame.

5.5. T.Encaps.L2.Red: Transit with reduced encaps of L2 frames

The T.Encaps.L2.Red behavior is an optimization of the T.Encaps.L2 behavior.

T.Encaps.L2.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

6. Operation

6.1. Counters

Any SRv6 capable node SHOULD implement the following set of combined counters (packets and bytes):

- CNT-1: Per local SID entry, traffic that matched that SID and was processed correctly.
- CNT-2: Per SRv6 Policy, traffic steered into it and processed correctly.

Furthermore, an SRv6 capable node maintains an aggregate counter CNT-3 tracking the IPv6 traffic that was received with a destination address matching a local interface address that is not a locally instantiated SID and the next-header is SRH with SL>0. We remind that this traffic is dropped as an interface address is not a local SID by default. A SID must be explicitly instantiated.

6.2. Flow-based hash computation

When a flow-based selection within a set needs to be performed, the source address, the destination address and the flow-label MUST be included in the flow-based hash.

This occurs when the destination address is updated, a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X, End.DX4, or End.DX6 are bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists [I-D.ietf-spring-segment-routing-policy].

Additionally, any transit router in an SRv6 domain MUST include the outer flow label in its ECMP load-balancing hash [RFC6437].

6.3. OAM

[I-D.ietf-6man-spring-srv6-oam] defines the OAM behavior for SRv6. This includes the definition of the SRH Flag 'O-bit', as well as additional OAM Endpoint behaviors.

7. Basic security for intra-domain deployment

The SRH Section 5.1 defines how a domain of trust can operate SRv6-based services for internal traffic while preventing any external traffic from accessing the internal SRv6-based services.

Future documents will detail inter-domain security mechanisms for SRv6 (e.g. how to allow external traffic to leverage internal SRv6 services).

8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

The specification of the SRv6 control-plane is outside the scope of this document.

We limit ourselves to a few important observations.

8.1. IGP

The End, End.T and End.X SIDs express topological behaviors and hence are expected to be signaled in the IGP together with the flavors PSP, USP and USD[I-D.ietf-lsr-isis-srv6-extensions].

The presence of SIDs in the IGP do not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the classic, non-SID-related, IGP information. Routing is not governed neither influenced in any way by a SID advertisement in the IGP.

These three SIDs provide important topological behaviors for the IGP to build FRR/TI-LFA solution and for TE processes relying on IGP LSDB to build SR policies.

8.2. BGP-LS

BGP-LS is expected to be the key service discovery protocol. Every node is expected to advertise via BGP-LS its SRv6 capabilities (e.g. how many SIDs it can insert as part of a T.Encaps behavior) and any locally instantiated SID [I-D.ietf-idr-bgpls-srv6-ext].

8.3. BGP IP/VPN/EVPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46, End.DX2, End.DX2V, End.DT2U and End.DT2M SIDs are expected to be signaled in BGP [I-D.ietf-bess-srv6-services].

8.4. Summary

The following table summarizes which SIDs are signaled in which signaling protocol.

| | IGP | BGP-LS | BGP IP/VPN/EVPN |
|-----------------------|-----|--------|-----------------|
| End (PSP, USP, USD) | X | X | |
| End.X (PSP, USP, USD) | X | X | |
| End.T (PSP, USP, USD) | X | X | |
| End.DX6 | X | X | X |
| End.DX4 | X | X | X |
| End.DT6 | X | X | X |
| End.DT4 | X | X | X |
| End.DT46 | X | X | X |
| End.DX2 | | X | X |
| End.DX2V | | X | X |
| End.DT2U | | X | X |
| End.DT2M | | X | X |
| End.B6.Encaps | | X | |
| End.B6.Encaps.Red | | X | |
| End.B6.BM | | X | |

Table 1: SRv6 locally instantiated SIDs signaling

The following table summarizes which transit capabilities are signaled in which signaling protocol.

| | IGP | BGP-LS | BGP IP/VPN/EVPN |
|-----------------|-----|--------|-----------------|
| T | | X | |
| T.Encaps | X | X | |
| T.Encaps.Red | X | X | |
| T.Encaps.L2 | | X | |
| T.Encaps.L2.Red | | X | |

Table 2: SRv6 transit behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SR policies.

For example, a BGP-LS advertisement of the T capability of node N would indicate that node N supports the basic transit behavior. The T.Encaps behavior would describe the capability of node N to perform a T.Encaps behavior, specifically it would describe how many SIDs could be inserted by N without significant performance degradation.

The reader should also remember that any specific instantiated SR policy is always assigned a Binding SID. They should remember that BSIDs are advertised in BGP-LS as shown in Table 1. Hence, it is normal that Table 2 only focuses on the generic capabilities related to T.Encaps as Table 1 advertises the specific instantiated BSID properties.

9. IANA Considerations

This document requests IANA to allocate a new IP Protocol Number value for "Ethernet" with the following definition: The value TBD1 in the Next Header field of an IPv6 header or any extension header indicates that the payload is Ethernet.

Additionally, this document requests the following new IANA registries:

- A new top-level registry "Segment-routing with IPv6 dataplane (SRv6) Parameters" to be created under IANA Protocol registries. This registry is being defined to serve as a top-level registry for keeping all other SRv6 sub-registries.
- A sub-registry "SRv6 Endpoint Behaviors" to be defined under top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry. This sub-registry maintains 16-bit identifiers for the SRv6 Endpoint behaviors. The range of the registry is 0-65535

(0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

| Range | Hex | Registration procedure | Notes |
|-------------|---------------|-------------------------------|---------|
| 0 | 0x0000 | Reserved | Invalid |
| 1-32767 | 0x0001-0x7FFF | Specification Required | |
| 32768-49151 | 0x8000-0xBFFF | Reserved for experimental use | |
| 49152-65534 | 0xC000-0xFFFE | Reserved for private use | Opaque |
| 65535 | 0xFFFF | Reserved | |

Table 3: SRv6 Endpoint Behaviors Registry

The initial registrations for the "Specification Required" portion of the sub-registry are as follows:

| Value | Hex | Endpoint behavior | Reference |
|-------|--------|------------------------|------------------------------------|
| 1 | 0x0001 | End (no PSP, no USP) | [This.ID] |
| 2 | 0x0002 | End with PSP | [This.ID] |
| 3 | 0x0003 | End with USP | [This.ID] |
| 4 | 0x0004 | End with PSP&USP | [This.ID] |
| 5 | 0x0005 | End.X (no PSP, no USP) | [This.ID] |
| 6 | 0x0006 | End.X with PSP | [This.ID] |
| 7 | 0x0007 | End.X with USP | [This.ID] |
| 8 | 0x0008 | End.X with PSP&USP | [This.ID] |
| 9 | 0x0009 | End.T (no PSP, no USP) | [This.ID] |
| 10 | 0x000A | End.T with PSP | [This.ID] |
| 11 | 0x000B | End.T with USP | [This.ID] |
| 12 | 0x000C | End.T with PSP&USP | [This.ID] |
| 13 | 0x000D | End.B6.Insert | [I-D.filsfils-spring-srv6-net-pgm- |

| | | | |
|----|------------|------------------------------|--|
| 14 | 0D 0x00 | End.B6.Encaps | insertion] [This.ID] |
| 15 | 0E 0x00 | End.BM | [This.ID] |
| 16 | 0F 0x00 | End.DX6 | [This.ID] |
| 17 | 10 0x00 | End.DX4 | [This.ID] |
| 18 | 11 0x00 | End.DT6 | [This.ID] |
| 19 | 12 0x00 | End.DT4 | [This.ID] |
| 20 | 13 0x00 | End.DT46 | [This.ID] |
| 21 | 14 0x00 | End.DX2 | [This.ID] |
| 22 | 15 0x00 | End.DX2V | [This.ID] |
| 23 | 16 0x00 | End.DT2U | [This.ID] |
| 24 | 17 0x00 | End.DT2M | [This.ID] |
| 25 | 18 0x00 | Reserved | [This.ID] |
| 26 | 19 0x00 | End.B6.Insert. | [I-D.filsfils-spring-srv6-net-pgm- insertion] |
| 27 | 1A 0x00 | Red | |
| 28 | 1B 0x00 | End.B6.Encaps. | [This.ID] |
| 29 | 1C 0x00 | Red | [This.ID] |
| 30 | 1D 0x00 | End with USD | [This.ID] |
| 31 | 1E 0x00 | End with PSP&USD | [This.ID] |
| 32 | 1F 0x00 | End with USP&USD | [This.ID] |
| 33 | 20 0x00 | End with PSP, USP & USD | [This.ID] |
| 34 | 21 0x00 | End.X with USD | [This.ID] |
| 35 | 22 0x00 | End.X with PSP&USD | [This.ID] |
| 36 | 23 0x00 | End.X with USP&USD | [This.ID] |
| 37 | 24 0x00 | End.X with PSP, USP & USD | [This.ID] |
| | 25 0x00 | End.T with USD | [This.ID] |
| | 26 0x00 | End.T with | [This.ID] |

| | | | |
|----|------------|-----------------------|-----------|
| 38 | 25 0x00 | PSP&USD End.T with | [This.ID] |
| 39 | 26 0x00 | USP&USD End.T with | [This.ID] |
| | 27 | PSP, USP & USD | |

Table 4: IETF - SRv6 Endpoint Behaviors

The SRv6 Endpoint Behavior numbers are maintained by the working group until the RFC is published. Note to the RFC Editor: Remove this paragraph before publication.

10. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Thierry Couture, Kris Michielsen, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulan, Satish Damodaran, John Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya and Saleem Hafeez.

11. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Lapishills Consulting Limited
Cyprus

Email: dirk@lapishills.com

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Ericsson
United States of America

Email: hani.elmalky@gmail.com

Prem Jonnalagadda
Barefoot Networks
United States of America

Email: prem@barefootnetworks.com

Milad Sharif
Barefoot Networks
United States of America

Email: msharif@barefootnetworks.com

David Lebrun
Google
Belgium

Email: dlebrun@google.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Arthi Ayyangar
Arista
United States of America

Email: arthi@arista.com

Satish Mynam
Innovium Inc.
United States of America

Email: smynam@innovium.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma
Juniper
Singapore

Email: mashao@juniper.net

Ahmed Bashandy
Individual
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Patrice Brissette
Cisco Systems, Inc.
Canada

Email: pbrisset@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America

Email: zali@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.
India

Email: ketant@cisco.com

12. References

12.1. Normative References

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [I-D.filsfils-spring-srv6-net-pgm-illustration]
Filsfils, C., Camarillo, P., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", draft-filsfils-spring-srv6-net-pgm-illustration-01 (work in progress), August 2019.

- [I-D.filsfils-spring-srv6-net-pgm-insertion]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
NET-PGM extension: Insertion", draft-filsfils-spring-srv6-
net-pgm-insertion-00 (work in progress), September 2019.
- [I-D.ietf-6man-spring-srv6-oam]
Ali, Z., Filsfils, C., Matsushima, S.,
daniel.voyer@bell.ca, d., and M. Chen, "Operations,
Administration, and Maintenance (OAM) in Segment Routing
Networks with IPv6 Data plane (SRv6)", draft-ietf-6man-
spring-srv6-oam-00 (work in progress), August 2019.
- [I-D.ietf-bess-srv6-services]
Dawra, G., Filsfils, C., Brissette, P., Agrawal, S.,
Leddy, J., daniel.voyer@bell.ca, d.,
daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R.,
Decraene, B., Matsushima, S., Zhuang, S., and J. Rabadan,
"SRv6 BGP based Overlay services", draft-ietf-bess-
srv6-services-00 (work in progress), October 2019.
- [I-D.ietf-idr-bgpls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Chen, M.,
daniel.bernier@bell.ca, d., and B. Decraene, "BGP Link
State Extensions for SRv6", draft-ietf-idr-bgpls-
srv6-ext-01 (work in progress), July 2019.
- [I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and
Z. Hu, "IS-IS Extension to Support Segment Routing over
IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-03
(work in progress), October 2019.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d.,
bogdanov@google.com, b., and P. Mattes, "Segment Routing
Policy Architecture", draft-ietf-spring-segment-routing-
policy-03 (work in progress), May 2019.
- [I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca,
d., Li, C., Decraene, B., Ma, S., Yadlapalli, C.,
Henderickx, W., and S. Salsano, "Service Programming with
Segment Routing", draft-ietf-spring-sr-service-
programming-00 (work in progress), October 2019.

[I-D.raza-spring-srv6-yang]

Raza, K., Rajamanickam, J., Liu, X., Hu, Z., Hussain, I., Shah, H., daniel.voyer@bell.ca, d., Elmalky, H., Matsushima, S., Horiba, K., and A. Abdelsalam, "YANG Data Model for SRv6 Base and Static", draft-raza-spring-srv6-yang-04 (work in progress), July 2019.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

John Leddy
Individual Contributor
United States of America

Email: john@leddy.net

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Satoru Matsushima
SoftBank
1-9-1, Higashi-Shimbashi, Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: July 2, 2021

C. Filsfils, Ed.
P. Camarillo, Ed.
Cisco Systems, Inc.
J. Leddy
Individual Contributor
D. Voyer
Bell Canada
S. Matsushima
SoftBank
Z. Li
Huawei Technologies
December 29, 2020

SRv6 Network Programming
draft-ietf-spring-srv6-network-programming-28

Abstract

The SRv6 Network Programming framework enables a network operator or an application to specify a packet processing program by encoding a sequence of instructions in the IPv6 packet header.

Each instruction is implemented on one or several nodes in the network and identified by an SRv6 Segment Identifier in the packet.

This document defines the SRv6 Network Programming concept and specifies the base set of SRv6 behaviors that enables the creation of interoperable overlays with underlay optimization.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 2, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 4 |
| 2. Terminology | 4 |
| 2.1. Requirements Language | 5 |
| 3. SRv6 SID | 5 |
| 3.1. SID Format | 6 |
| 3.2. SID Allocation within an SR domain | 7 |
| 3.3. SID Reachability | 9 |
| 4. SR Endpoint Behaviors | 10 |
| 4.1. End: Endpoint | 12 |
| 4.1.1. Upper-Layer Header | 12 |
| 4.2. End.X: Layer-3 Cross-Connect | 13 |
| 4.3. End.T: Specific IPv6 Table Lookup | 14 |
| 4.4. End.DX6: Decapsulation and IPv6 Cross-Connect | 14 |
| 4.5. End.DX4: Decapsulation and IPv4 Cross-Connect | 15 |
| 4.6. End.DT6: Decapsulation and Specific IPv6 Table Lookup | 16 |
| 4.7. End.DT4: Decapsulation and Specific IPv4 Table Lookup | 17 |
| 4.8. End.DT46: Decapsulation and Specific IP Table Lookup | 18 |
| 4.9. End.DX2: Decapsulation and L2 Cross-Connect | 19 |
| 4.10. End.DX2V: Decapsulation and VLAN L2 Table Lookup | 20 |
| 4.11. End.DT2U: Decapsulation and Unicast MAC L2 Table Lookup | 21 |
| 4.12. End.DT2M: Decapsulation and L2 Table Flooding | 22 |
| 4.13. End.B6.Encaps: Endpoint Bound to an SRv6 Policy w/ Encaps | 22 |
| 4.14. End.B6.Encaps.Red: End.B6.Encaps with Reduced SRH | 24 |
| 4.15. End.BM: Endpoint Bound to an SR-MPLS Policy | 24 |
| 4.16. Flavors | 25 |
| 4.16.1. PSP: Penultimate Segment Pop of the SRH | 25 |
| 4.16.2. USP: Ultimate Segment Pop of the SRH | 28 |
| 4.16.3. USD: Ultimate Segment Decapsulation | 28 |
| 5. SR Policy Headend Behaviors | 29 |
| 5.1. H.Encaps: SR Headend with Encapsulation in an SRv6 Policy | 30 |
| 5.2. H.Encaps.Red: H.Encaps with Reduced Encapsulation | 31 |

| | | |
|---------|--|----|
| 5.3. | H.Encaps.L2: H.Encaps Applied to Received L2 Frames . . . | 31 |
| 5.4. | H.Encaps.L2.Red: H.Encaps.Red Applied to Received L2 frames | 31 |
| 6. | Counters | 32 |
| 7. | Flow-based Hash Computation | 32 |
| 8. | Control Plane | 32 |
| 8.1. | IGP | 33 |
| 8.2. | BGP-LS | 33 |
| 8.3. | BGP IP/VPN/EVPN | 33 |
| 8.4. | Summary | 33 |
| 9. | Security Considerations | 35 |
| 10. | IANA Considerations | 35 |
| 10.1. | Ethernet Next Header Type | 35 |
| 10.2. | SRv6 Endpoint Behaviors Registry | 36 |
| 10.2.1. | Initial Registrations | 36 |
| 11. | Acknowledgements | 38 |
| 12. | Contributors | 38 |
| 13. | References | 41 |
| 13.1. | Normative References | 41 |
| 13.2. | Informative References | 42 |
| | Authors' Addresses | 43 |

1. Introduction

Segment Routing [RFC8402] leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the Segment List to any complex user-defined behavior. Network programming combines segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document defines the SRv6 Network Programming concept and specifies the main segment routing behaviors to enable the creation of interoperable overlays with underlay optimization.

The companion document [I-D.filsfils-spring-srv6-net-pgm-illustration] illustrates the concepts defined in this document.

Familiarity with the Segment Routing Header [RFC8754] is expected.

2. Terminology

The following terms used within this document are defined in [RFC8402]: Segment Routing, SR Domain, Segment ID (SID), SRv6, SRv6 SID, SR Policy, Prefix-SID, and Adj-SID.

The following terms used within this document are defined in [RFC8754]: SRH, SR Source Node, Transit Node, SR Segment Endpoint Node, Reduced SRH, Segments Left and Last Entry.

SL: The Segments Left field of the SRH

FIB: Forwarding Information Base. A FIB lookup is a lookup in the forwarding table.

SA: Source Address

DA: Destination Address

SRv6 SID function: The function part of the SID is an opaque identification of a local behavior bound to the SID. It is formally defined in Section 3.1 of this document.

SRv6 Segment Endpoint behavior: A packet processing behavior executed at an SRv6 Segment Endpoint Node. Section 4 of this document defines SRv6 Segment Endpoint behaviors related to traffic-engineering and

overlay use-cases. Other behaviors (e.g. service programming) are outside the scope of this document.

An SR Policy is resolved to a SID list. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Address is DA, and next-header is SRH.
- SRH with SID list <S1, S2, S3> with Segments Left = SL.
- Note the difference between the <> and () symbols: <S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID to traverse. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of the detailed packet behavior, the (S3, S2, S1; SL) notation is more convenient.
- The payload of the packet is omitted.

Per-VRF VPN label: a single label for the entire VRF that is shared by all routes from that VRF ([RFC4364] Section 4.3.2)

Per-CE VPN label: a single label for each attachment circuit that is shared by all routes with the same "outgoing attachment circuit" ([RFC4364] Section 4.3.2)

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. SRv6 SID

RFC8402 defines an SRv6 Segment Identifier as an IPv6 address explicitly associated with the segment.

When an SRv6 SID is in the Destination Address field of an IPv6 header of a packet, it is routed through Transit Nodes in an IPv6 network as an IPv6 address.

Its processing is defined in [RFC8754] section 4.3 and reproduced here as a reminder.

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packet's destination address. This lookup can return any of the following:

- * A FIB entry that represents a locally instantiated SRv6 SID
- * A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- * A FIB entry that represents a nonlocal route
- * No Match

Section 4 of this document defines a new set of SRv6 SID behaviors in addition to that defined in [RFC8754] Section 4.3.1.

3.1. SID Format

This document defines an SRv6 SID as consisting of LOC:FUNCT:ARG, where a locator (LOC) is encoded in the L most significant bits of the SID, followed by F bits of function (FUNCT) and A bits of arguments (ARG). L, the locator length, is flexible, and an operator is free to use the locator length of their choice. F and A may be any value as long as $L+F+A \leq 128$. When $L+F+A$ is less than 128 then the remaining bits of the SID MUST be zero.

A locator may be represented as B:N where B is the SRv6 SID block (IPv6 prefix allocated for SRv6 SIDs by the operator) and N is the identifier of the parent node instantiating the SID.

When the LOC part of the SRv6 SIDs is routable, it leads to the node which instantiates the SID.

The FUNCT is an opaque identification of a local behavior bound to the SID.

The term "function" refers to the bit-string in the SRv6 SID. The term "behavior" identifies the behavior bound to the SID. Some behaviors are defined in Section 4 of this document.

An SRv6 Segment Endpoint Behavior may require additional information for its processing (e.g. related to the flow or service). This information may be encoded in the ARG bits of the SID.

In such a case, the semantics and format of the ARG bits are defined as part of the SRv6 endpoint behavior specification.

The ARG value of a routed SID SHOULD remain constant among packets in a given flow. Varying ARG values among packets in a flow may result in different ECMP hashing and cause re-ordering.

3.2. SID Allocation within an SR domain

Locators are assigned consistent with IPv6 infrastructure allocation. For example, a network operator may:

- o Assign block B::/48 to the SR domain
- o Assign a unique B:N::/64 block to each SRv6-enabled node in the domain

As an example, one mobile service provider has commercially deployed SRv6 across more than 1000 commercial routers and 1800 whitebox routers. All these devices are enabled for SRv6 and advertise SRv6 SIDs. The provider historically deployed IPv6 and assigned infrastructure addresses from ULA space [RFC4193]. They specifically allocated three /48 prefixes (Country X, Country Y, Country Z) to support their SRv6 infrastructure. From those /48 prefixes each router was assigned a /64 prefix from which all SIDs of that router are allocated.

In another example, a large mobile and fixed-line service provider has commercially deployed SRv6 in their country-wide network. This provider is assigned a /20 prefix by an RIR (Regional Internet Registry). They sub-allocated a few /48 prefixes to their infrastructure to deploy SRv6. Each router is assigned a /64 prefix from which all SIDs of that router are allocated.

IPv6 address consumption in both these examples is minimal, representing less than one billionth and one millionth of the available address space, respectively.

A service provider receiving the current minimum allocation of a /32 from an RIR may assign a /48 prefix to their infrastructure deploying

SRv6, and subsequently allocate /64 prefixes for SIDs at each SRv6 node. The /48 assignment is one sixty-five thousandth ($1/2^{16}$) of the usable IPv6 address space available for assignment by the provider.

When an operator instantiates a SID at a node, they specify a SID value B:N:FUNCT and the behavior bound to the SID using one of the SRv6 Endpoint Behavior codepoint of the registry defined in this document (see Table 4).

The node advertises the SID, B:N:FUNCT, in the control-plane (see Section 8) together with the SRv6 Endpoint Behavior codepoint identifying the behavior of the SID.

An SR Source Node cannot infer the behavior by examination of the FUNCT value of a SID.

Therefore, the SRv6 Endpoint Behavior codepoint is advertised along with the SID in the control plane.

An SR Source Node uses the SRv6 Endpoint Behavior codepoint to map the received SID (B:N:FUNCT) to a behavior.

An SR Source Node selects a desired behavior at an advertising node by selecting the SID (B:N:FUNCT) advertised with the desired behavior.

As an example, a network operator may:

- o Assign an SRv6 SID block 2001:db8:bbbb::/48 from their in-house operation block for their SRv6 infrastructure
- o Assign an SRv6 Locator 2001:db8:bbbb:3::/64 to one particular router, for example Router 3, in their SR Domain
- o At Router 3, within the locator 2001:db8:bbbb:3::/64, the network operator or the router performs dynamic assignment for:
 - * Function 0x0100 associated with the behavior End.X (Endpoint with cross-connect) between router 3 and its connected neighbor router, for example Router 4. This function is encoded as 16-bit value and has no arguments (F=16, A=0). This SID is advertised in the control plane as 2001:db8:bbbb:3:100:: with SRv6 Endpoint Behavior codepoint value of 5.
 - * Function 0x0101 associated with the behavior End.X (Endpoint with cross-connect) between router 3 and its connected neighbor

router, for example Router 2. This function is encoded as 16-bit value and has no arguments (F=16, A=0). This SID is advertised in the control plane as 2001:db8:bbbb:3:101:: with SRv6 Endpoint Behavior codepoint value of 5.

These examples do not preclude any other IPv6 addressing allocation scheme.

3.3. SID Reachability

Most often, the node N would advertise IPv6 prefix(es) matching the LOC parts covering its SIDs or shorter-mask prefix. The distribution of these advertisements and calculation of their reachability are specific to the routing protocol and are outside of the scope of this document.

An SRv6 SID is said to be routed if its SID belongs to an IPv6 prefix advertised via a routing protocol. An SRv6 SID that does not fulfill this condition is non-routed.

Let's provide a classic illustration:

Node N is configured explicitly with two SIDs: 2001:db8:b:1:100:: and 2001:db8:b:2:101::.

The network learns about a path to 2001:db8:b:1::/64 via the IGP and hence a packet destined to 2001:db8:b:1:100:: would be routed up to N. The network does not learn about a path to 2001:db8:b:2::/64 via the IGP and hence a packet destined to 2001:db8:b:2:101:: would not be routed up to N.

A packet could be steered through a non-routed SID 2001:db8:b:2:101:: by using a SID list <...,2001:db8:b:1:100::,2001:db8:b:2:101::,...> where the non-routed SID is preceded by a routed SID to the same node. A packet could also be steered to a node instantiating a non-routed SID by preceding it in the SID-list with an Adjacency SID to that node. Routed and non-routed SRv6 SIDs are the SRv6 instantiation of global and local segments, respectively [RFC8402].

4. SR Endpoint Behaviors

Following is a set of well-known behaviors that can be associated with a SID.

| | |
|-------------------|--|
| End | Endpoint function |
| End.X | The SRv6 instantiation of a Prefix SID [RFC8402] |
| End.T | Endpoint with Layer-3 cross-connect |
| End.DX6 | The SRv6 instantiation of an Adj SID [RFC8402] |
| End.DX4 | Endpoint with specific IPv6 table lookup |
| End.DT6 | Endpoint with decapsulation and IPv6 cross-connect |
| End.DT4 | e.g. IPv6-L3VPN (equivalent to per-CE VPN label) |
| End.DT46 | Endpoint with decaps and IPv4 cross-connect |
| End.DX2 | e.g. IPv4-L3VPN (equivalent to per-CE VPN label) |
| End.DX2V | Endpoint with decapsulation and IPv6 table lookup |
| End.DT2U | e.g. IPv6-L3VPN (equivalent to per-VRF VPN label) |
| End.DT2M | Endpoint with decapsulation and IPv4 table lookup |
| End.B6.Encaps | e.g. IPv4-L3VPN (equivalent to per-VRF VPN label) |
| End.B6.Encaps.Red | Endpoint with decapsulation and IP table lookup |
| End.BM | e.g. IP-L3VPN (equivalent to per-VRF VPN label) |
| | Endpoint with decapsulation and L2 cross-connect |
| | e.g. L2VPN use-case |
| | Endpoint with decaps and VLAN L2 table lookup |
| | e.g. EVPN Flexible cross-connect use-case |
| | Endpoint with decaps and unicast MAC L2 table lookup |
| | e.g. EVPN Bridging unicast use-case |
| | Endpoint with decapsulation and L2 table flooding |
| | e.g. EVPN Bridging BUM use-case with ESI filtering |
| | Endpoint bound to an SRv6 policy with encapsulation |
| | SRv6 instantiation of a Binding SID |
| | End.B6.Encaps with reduced SRH |
| | SRv6 instantiation of a Binding SID |
| | Endpoint bound to an SR-MPLS Policy |
| | SRv6 instantiation of an SR-MPLS Binding SID |

The list is not exhaustive. In practice, any behavior can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex processing on the packet, provided there is a behavior codepoint allocated for the processing.

When an SRv6-capable node (N) receives an IPv6 packet whose destination address matches a FIB entry that represents a locally instantiated SRv6 SID (S), the IPv6 header chain is processed as defined in Section 4 of [RFC8200]. For SRv6 SIDs associated with an Endpoint Behavior defined in this document, the SRH and Upper-layer Header are processed as defined in the following subsections.

The pseudocode describing these behaviors details local processing at a node. An implementation of the pseudocode is compliant as long as the externally observable wire protocol is as described by the pseudocode.

Section 4.16 defines flavors of some of these behaviors.

Section 10.2 of this document defines the IANA Registry used to maintain all these behaviors as well as future ones defined in other documents.

4.1. End: Endpoint

The Endpoint behavior ("End" for short) is the most basic behavior. It is the instantiation of a Prefix-SID [RFC8402].

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
        header in the packet, whose type is identified by
        the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S16. }
```

Notes:

The End behavior operates on the same FIB table (i.e. identified by VRF or L3 relay id) associated to the packet. Hence the FIB lookup on line S15 is done in the same FIB table as the ingress interface.

4.1.1. Upper-Layer Header

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End SID, N does:

```
S01. If (Upper-Layer Header type is allowed by local configuration) {
S02.   Proceed to process the Upper-layer Header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address,
       Code 4 (SR Upper-layer Header Error),
       Pointer set to the offset of the Upper-layer Header,
       Interrupt packet processing and discard the packet.
S05 }
```

Allowing processing of specific Upper-Layer Headers types is useful for OAM. As an example, an operator might permit pinging of SIDs. To do this they may enable local configuration to allow Upper-layer Header type 58 (ICMPv6).

It is RECOMMENDED that an implementation of local configuration only allows Upper-layer Header processing of types that do not result in the packet being forwarded (e.g. ICMPv6).

4.2. End.X: Layer-3 Cross-Connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" behavior (End.X for short) is a variant of the End behavior.

It is the SRv6 instantiation of an Adjacency-SID [RFC8402] and its main use is for traffic-engineering policies.

Any SID instance of this behavior is associated with a set, J, of one or more Layer-3 adjacencies.

When N receives a packet destined to S and S is a local End.X SID, the line S15 from the End processing is replaced by the following:

```
S15.   Submit the packet to the IPv6 module for transmission
       to the new destination via a member of J
```

Notes:

S15. If the set J contains several L3 adjacencies, then one element of the set is selected based on a hash of the packet's header (see Section 7).

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N might allocate up to 11 End.X local SIDs: one for the bundle itself and then up to one for each Layer-2 member link. The flows steered using the End.X SID corresponding to the bundle itself get load balanced across the member links via hashing while the flows steered using the End.X SID corresponding to a member link get steered over that specific member link alone.

When the End.X behavior is associated with a BGP Next-Hop, it is the SRv6 instantiation of the BGP Peering Segments [RFC8402].

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End.X SID, process the packet as per Section 4.1.1.

4.3. End.T: Specific IPv6 Table Lookup

The "Endpoint with specific IPv6 table lookup" behavior (End.T for short) is a variant of the End behavior.

The End.T behavior is used for multi-table operation in the core. For this reason, an instance of the End.T behavior is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.T SID, the line S15 from the End processing is replaced by the following:

- S15.1. Set the packet's associated FIB table to T
- S15.2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End.T SID, process the packet as per Section 4.1.1.

4.4. End.DX6: Decapsulation and IPv6 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" behavior (End.DX6 for short) is a variant of the End.X behavior.

One of the applications of the End.DX6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress

Provider Edge (PE) is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX6 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv6 adjacencies J.

When N receives a packet destined to S and S is a local End.DX6 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX6 SID, N does:

```
S01. If (Upper-Layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IPv6 packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

S03. If the End.DX6 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header (see Section 7).

4.5. End.DX4: Decapsulation and IPv4 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" behavior (End.DX4 for short) is a variant of the End.X behavior.

One of the applications of the End.DX4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX4 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv4 adjacencies J.

When N receives a packet destined to S and S is a local End.DX4 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX4 SID, N does:

```
S01. If (Upper-Layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IPv4 packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

S03. If the End.DX4 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header (see Section 7).

4.6. End.DT6: Decapsulation and Specific IPv6 Table Lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" behavior (End.DT6 for short) is a variant of the End.T behavior.

One of the applications of the End.DT6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case an End.DT6 supports the equivalent of an IPv6inIPv6 decapsulation (without VPN/tenant implication).

The End.DT6 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.DT6 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT6 SID, N does:

```
S01. If (Upper-Layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

4.7. End.DT4: Decapsulation and Specific IPv4 Table Lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" behavior (End.DT4 for short) is a variant of the End.T behavior.

One of the applications of the End.DT4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case an End.DT4 supports the equivalent of an IPv4inIPv6 decapsulation (without VPN/tenant implication).

The End.DT4 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T.

When N receives a packet destined to S and S is a local End.DT4 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT4 SID, N does:

```
S01. If (Upper-Layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv4 FIB lookup and
        transmission to the new destination
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

4.8. End.DT46: Decapsulation and Specific IP Table Lookup

The "Endpoint with decapsulation and specific IP table lookup" behavior (End.DT46 for short) is a variant of the End.DT4 and End.DT6 behavior.

One of the applications of the End.DT46 behavior is the L3VPN use-case where a FIB lookup in a specific IP tenant table at the egress PE is required. This is equivalent to single per-VRF VPN label (for IPv4 and IPv6) in MPLS[RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case an End.DT46 supports the equivalent of an IPinIPv6 decapsulation(without VPN/tenant implication).

The End.DT46 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T4 and an IPv6 FIB table T6.

When N receives a packet destined to S and S is a local End.DT46 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT46 SID, N does:

```
S01. If (Upper-layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T4
S04.   Submit the packet to the egress IPv4 FIB lookup and
        transmission to the new destination
S05. } Else if (Upper-layer Header type == 41(IPv6) ) {
S06.   Remove the outer IPv6 Header with all its extension headers
S07.   Set the packet's associated FIB table to T6
S08.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S09. } Else {
S10.   Process as per Section 4.1.1
S11. }
```

4.9. End.DX2: Decapsulation and L2 Cross-Connect

The "Endpoint with decapsulation and Layer-2 cross-connect to an outgoing L2 interface (OIF)" (End.DX2 for short) is a variant of the endpoint behavior.

One of the applications of the End.DX2 behavior is the L2VPN [RFC4664] / EVPN VPWS [RFC7432] [RFC8214] use-case.

The End.DX2 SID MUST be the last segment in a SR Policy, and it is associated with one outgoing interface I.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX2 SID, N does:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the Ethernet frame to the OIF I
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet [IEEE.802.3_2018] (see Section 10.1).
S03. An End.DX2 behavior could be customized to expect a specific IEEE header (e.g. VLAN tag) and rewrite the egress IEEE header before forwarding on the outgoing interface.

Note that an End.DX2 SID may also be associated with a bundle of outgoing interfaces.

4.10. End.DX2V: Decapsulation and VLAN L2 Table Lookup

The "Endpoint with decapsulation and specific VLAN table lookup" behavior (End.DX2V for short) is a variant of the End.DX2 behavior.

One of the applications of the End.DX2V behavior is the EVPN Flexible cross-connect use-case. The End.DX2V behavior is used to perform a lookup of the Ethernet frame VLANs in a particular L2 table. Any SID instance of this behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DX2 SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is modified as follows:

S03. Lookup the exposed VLANs in L2 table T, and forward via the matched table entry.

Notes:

S03. An End.DX2V behavior could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

4.11. End.DT2U: Decapsulation and Unicast MAC L2 Table Lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" behavior (End.DT2U for short) is a variant of the End behavior.

One of the applications of the End.DT2U behavior is the EVPN Bridging unicast [RFC7432]. Any SID instance of the End.DT2U behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2U SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Learn the exposed MAC Source Address in L2 Table T
S04.   Lookup the exposed MAC Destination Address in L2 Table T
S05.   If (matched entry in T) {
S06.     Forward via the matched table T entry
S07.   } Else {
S08.     Forward via all L2 OIFs entries in table T
S09.   }
S10. } Else {
S11.   Process as per Section 4.1.1
S12. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet (see Section 10.1).

S03. In EVPN [RFC7432], the learning of the exposed MAC Source Address is done via control plane. In L2VPN VPLS [RFC4761] [RFC4762] reachability is obtained by standard learning bridge functions in the data plane.

4.12. End.DT2M: Decapsulation and L2 Table Flooding

The "Endpoint with decapsulation and specific L2 table flooding" behavior (End.DT2M for short) is a variant of the End.DT2U behavior.

Two of the applications of the End.DT2M behavior are the EVPN Bridging of broadcast, unknown and multicast (BUM) traffic with Ethernet Segment Identifier (ESI) filtering [RFC7432] and the EVPN ETREE [RFC8317] use-cases.

Any SID instance of this behavior is associated with a L2 table T. The behavior also takes an argument: "Arg.FE2". This argument provides a local mapping to ESI for split-horizon filtering of the received traffic to exclude specific OIF (or set of OIFs) from L2 table T flooding. The allocation of the argument values is local to the SR Endpoint Node instantiating this behavior and the signaling of the argument to other nodes for the EVPN functionality occurs via control plane.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2M SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Learn the exposed MAC Source Address in L2 Table T
S04.   Forward via all L2OIFs excluding those associated by the
       identifier Arg.FE2
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet (see Section 10.1).

S03. In EVPN [RFC7432], the learning of the exposed MAC Source Address is done via control plane. In L2VPN VPLS [RFC4761] [RFC4762] reachability is obtained by standard learning bridge functions in the data plane.

4.13. End.B6.Encaps: Endpoint Bound to an SRv6 Policy w/ Encaps

This is a variation of the End behavior.

One of its applications is to express scalable traffic-engineering policies across multiple domains. It is one of the SRv6 instantiations of a Binding SID [RFC8402].

Any SID instance of this behavior is associated with an SR Policy B and a source address A.

When N receives a packet whose IPv6 DA is S and S is a local End.B6.Encaps SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
        header in the packet, whose type is identified by
        the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Push a new IPv6 header with its own SRH containing B
S16.   Set the outer IPv6 SA to A
S17.   Set the outer IPv6 DA to the first SID of B
S18.   Set the outer Payload Length, Traffic Class, Flow Label,
        Hop Limit and Next-Header fields
S19.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S20. }
```

Notes:

S15. The SRH MAY be omitted when the SRv6 Policy B only contains one SID and there is no need to use any flag, tag or TLV.

S18. The Payload Length, Traffic Class, Hop Limit and Next-Header fields are set as per [RFC2473]. The Flow Label is computed as per [RFC6437].

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID, process the packet as per Section 4.1.1.

4.14. End.B6.Encaps.Red: End.B6.Encaps with Reduced SRH

This is an optimization of the End.B6.Encaps behavior.

End.B6.Encaps.Red reduces the size of the SRH by one SID by excluding the first SID in the SRH of the new IPv6 header. Thus, the first segment is only placed in the IPv6 Destination Address of the new IPv6 header and the packet is forwarded according to it.

The SRH Last Entry field is set as defined in Section 4.1.1 of [RFC8754].

The SRH MAY be omitted when the SRv6 Policy only contains one SID and there is no need to use any flag, tag or TLV.

4.15. End.BM: Endpoint Bound to an SR-MPLS Policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End behavior.

The End.BM behavior is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing. This is an SRv6 instantiation of an SR-MPLS Binding SID [RFC8402].

Any SID instance of this behavior is associated with an SR-MPLS Policy B.

When N receives a packet whose IPv6 DA is S and S is a local End.BM SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
           header in the packet, whose type is identified by
           the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.

S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Push the MPLS label stack for B
S16.   Submit the packet to the MPLS engine for transmission
S17. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.BM SID, process the packet as per Section 4.1.1.

4.16. Flavors

The Penultimate Segment Pop of the SRH (PSP), Ultimate Segment Pop of the SRH (USP) and Ultimate Segment Decapsulation (USD) flavors are variants of the End, End.X and End.T behaviors. The End, End.X and End.T behaviors can support these flavors either individually or in combinations.

4.16.1. PSP: Penultimate Segment Pop of the SRH

4.16.1.1. Guidelines

SR Segment Endpoint Nodes advertise the SIDs instantiated on them via control plane protocols as described in Section 8. Different behavior ids are allocated for flavored and unflavored SIDs (see Table 4).

An SR Segment Endpoint Node that offers both PSP and non-PSP flavored behavior advertises them as two different SIDs.

The SR Segment Endpoint Node only advertises the PSP flavor if the operator enables this capability at the node.

The PSP operation is deterministically controlled by the SR Source Node.

A PSP-flavored SID is used by the Source SR Node when it needs to instruct the penultimate SR Segment Endpoint Node listed in the SRH to remove the SRH from the IPv6 header.

4.16.1.2. Definition

SR Segment Endpoint Nodes receive the IPv6 packet with the Destination Address field of the IPv6 Header equal to its SID address.

A penultimate SR Segment Endpoint Node is one that, as part of the SID processing, copies the last SID from the SRH into the IPv6 Destination Address and decrements the Segments Left value from one to zero.

The PSP operation only takes place at a penultimate SR Segment Endpoint Node and does not happen at any Transit Node. When a SID of PSP-flavor is processed at a non-penultimate SR Segment Endpoint Node, the PSP behavior is not performed as described in the pseudocode below since Segments Left would not be zero.

The SRH processing of the End, End.X and End.T behaviors are modified: after the instruction "S14. Update IPv6 DA with Segment List[Segments Left]" is executed, the following instructions must be executed as well:

```
S14.1.  If (Segments Left == 0) {
S14.2.      Update the Next Header field in the preceding header to the
           Next Header value from the SRH
S14.3.      Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
S14.4.      Remove the SRH from the IPv6 extension header chain
S14.5.  }
```

The usage of PSP does not increase the MTU of the IPv6 packet and hence does not have any impact on the PMTU discovery mechanism.

As a reminder, [RFC8754] defines in section 5 the SR Deployment Model within the SR Domain [RFC8402]. Within this framework, the Authentication Header (AH) is not used to secure the SRH as described

in Section 7.5 of [RFC8754]. Hence, the discussion of applicability of PSP along with AH usage is beyond the scope of this document.

In the context of this specification, the End, End.X and End.T behaviors with PSP do not contravene Section 4 of [RFC8200] because the destination address of the incoming packet is the address of the node executing the behavior.

4.16.1.3. Use-case

One use-case for the PSP functionality is streamlining the operation of an egress border router.

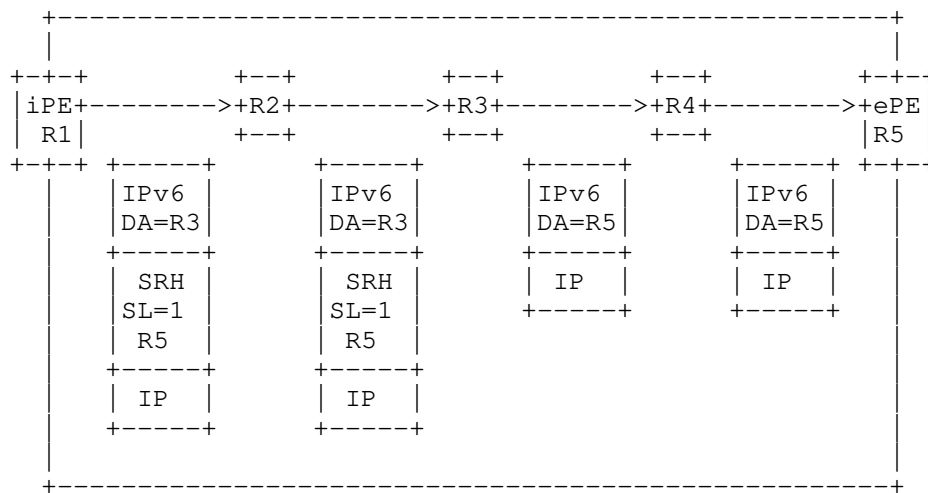


Figure 1: PSP use-case topology

In the above illustration, for a packet sent from iPE to ePE, node R3 is an intermediate traffic engineering waypoint and is the penultimate segment endpoint router; the node that copies the last segment from the SRH into the IPv6 Destination Address and decrements segments left to 0. The SDN controller knows that no other node after R3 needs to inspect the SRH, and it instructs R3 to remove the exhausted SRH from the packet by using a PSP-flavored SID.

The benefits for the egress PE are straightforward:

- as part of the decapsulation process the egress PE is required to parse and remove fewer bytes from the packet.
- if a lookup on an upper-layer IP header is required (e.g. per-VRF VPN), the header is more likely to be within the memory accessible

to the lookup engine in the forwarding ASIC (Application-specific integrated circuit).

4.16.2. USP: Ultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
S02.    If (Segments Left == 0) {
S03.1.    Update the Next Header field in the preceding header to the
           Next Header value of the SRH
S03.2.    Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
S03.3.    Remove the SRH from the IPv6 extension header chain
S03.4.    Proceed to process the next header in the packet
S04.    }
```

One of the applications of the USP flavor is when a packet with an SRH is destined to an application on hosts with smartNICs implementing SRv6. The USP flavor is used to remove the consumed SRH from the extension header chain before sending the packet to the host.

4.16.3. USD: Ultimate Segment Decapsulation

The Upper-layer header processing of the End, End.X and End.T behaviors are modified as follows:

```
End:
S01. If (Upper-layer Header type == 41(IPv6) ) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Submit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
S04. } Else if (Upper-layer Header type == 4(IPv4) ) {
S05.    Remove the outer IPv6 Header with all its extension headers
S06.    Submit the packet to the egress IPv4 FIB lookup and
           transmission to the new destination
S07. Else {
S08.    Process as per Section 4.1.1
S09. }
```

```

End.T:
S01. If (Upper-layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv6 FIB lookup and
       transmission to the new destination
S05. } Else if (Upper-layer Header type == 4(IPv4) ) {
S06.   Remove the outer IPv6 Header with all its extension headers
S07.   Set the packet's associated FIB table to T
S08.   Submit the packet to the egress IPv4 FIB lookup and
       transmission to the new destination
S09. Else {
S10.   Process as per Section 4.1.1
S11. }

End.X:
S01. If (Upper-layer Header type == 41(IPv6) ||
       Upper-layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IP packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }

```

One of the applications of the USD flavor is the case of TI-LFA in P routers with encapsulation. The USD flavor allows the last Segment Endpoint Node in the repair path list to decapsulate the IPv6 header added at the TI-LFA Point of Local Repair and forward the inner packet.

5. SR Policy Headend Behaviors

This section describes a set of SR Policy Headend [RFC8402] behaviors.

| | |
|-----------------|--|
| H.Encaps | SR Headend Behavior with Encapsulation in an SR Policy |
| H.Encaps.Red | H.Encaps with Reduced Encapsulation |
| H.Encaps.L2 | H.Encaps Applied to Received L2 Frames |
| H.Encaps.L2.Red | H.Encaps.Red Applied to Received L2 Frames |

This list is not exhaustive and future documents may define additional behaviors.

5.1. H.Encaps: SR Headend with Encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2) (B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

Node N is configured with an IPv6 Address T (e.g. assigned to its loopback).

N steers the transit packets P1 and P2 into an SR Policy with a Source Address T and a Segment list <S1, S2, S3>.

The H.Encaps encapsulation behavior is defined as follows:

- S01. Push an IPv6 header with its own SRH
- S02. Set outer IPv6 SA = T and outer IPv6 DA to the first SID in the segment list
- S03. Set outer Payload Length, Traffic Class, Hop Limit and Flow Label fields
- S04. Set the outer Next-Header value
- S05. Decrement inner IPv6 Hop Limit or IPv4 TTL
- S06. Submit the packet to the IPv6 module for transmission to S1

Note:

S03: As described in [RFC2473] and [RFC6437].

After the H.Encaps behavior, P1' and P2' respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)
- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The received packet is encapsulated unmodified (with the exception of the IPv4 TTL or IPv6 Hop Limit that is decremented as described in [RFC2473]).

The H.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployments. It may be also used for TI-LFA [I-D.ietf-rtgwg-segment-routing-ti-lfa] at the point of local repair.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.2. H.Encaps.Red: H.Encaps with Reduced Encapsulation

The H.Encaps.Red behavior is an optimization of the H.Encaps behavior.

H.Encaps.Red reduces the length of the SRH by excluding the first SID in the SRH of the pushed IPv6 header. The first SID is only placed in the Destination Address field of the pushed IPv6 header.

After the H.Encaps.Red behavior, P1' and P2' respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)
- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.3. H.Encaps.L2: H.Encaps Applied to Received L2 Frames

The H.Encaps.L2 behavior encapsulates a received Ethernet [IEEE.802.3_2018] frame and its attached VLAN header, if present, in an IPv6 packet with an SRH. The Ethernet frame becomes the payload of the new IPv6 packet.

The Next Header field of the SRH MUST be set to 143.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

The encapsulating node MUST remove the preamble (if any) and frame check sequence (FCS) from the Ethernet frame upon encapsulation and the decapsulating node MUST regenerate, as required, the preamble and FCS before forwarding Ethernet frame.

5.4. H.Encaps.L2.Red: H.Encaps.Red Applied to Received L2 frames

The H.Encaps.L2.Red behavior is an optimization of the H.Encaps.L2 behavior.

H.Encaps.L2.Red reduces the length of the SRH by excluding the first SID in the SRH of the pushed IPv6 header. The first SID is only places in the Destination Address field of the pushed IPv6 header.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

6. Counters

A node supporting this document SHOULD implement a pair of traffic counters (one for packets and one for bytes) per local SID entry, for traffic that matched that SID and was processed successfully (i.e. packets which generate ICMP Error Messages or are dropped are not counted). The retrieval of these counters from MIB, NETCONF/YANG or any other data structure is outside the scope of this document.

7. Flow-based Hash Computation

When a flow-based selection within a set needs to be performed, the IPv6 Source Address, the IPv6 Destination Address and the IPv6 Flow Label of the outer IPv6 header MUST be included in the flow-based hash.

This occurs when a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X, End.DX4, or End.DX6 are bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists.

Additionally, any transit router in an SRv6 domain includes the outer flow label in its ECMP flow-based hash [RFC6437].

8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

While not necessary for an SDN control plane, the remainder of this section provides a high-level illustrative overview of how control-plane protocols may be involved with SRv6. Their specification is outside the scope of this document.

8.1. IGP

The End, End.T and End.X SIDs express topological behaviors and hence are expected to be signaled in the IGP together with the flavors PSP, USP and USD. The IGP should also advertise the maximum SRv6 SID depth (MSD) capability of the node for each type of SRv6 operation - in particular, the SR source (e.g. H.Encaps), intermediate endpoint (e.g. End, End.X) and final endpoint (e.g. End.DX4, End.DT6) behaviors. These capabilities are factored in by an SR Source Node (or a controller) during the SR Policy computation.

The presence of SIDs in the IGP does not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the non-SID-related IGP prefix reachability information that includes locators. Routing is neither governed nor influenced in any way by a SID advertisement in the IGP.

These SIDs provide important topological behaviors for the IGP to build FRR solutions based on TI-LFA [I-D.ietf-rtgwg-segment-routing-ti-lfa] and for TE processes relying on IGP topology database to build SR policies.

8.2. BGP-LS

BGP-LS provides the functionality for topology discovery that includes the SRv6 capabilities of the nodes, their locators and locally instantiated SIDs. This enables controllers or applications to build an inter-domain topology that can be used for computation of SR Policies using the SRv6 SIDs.

8.3. BGP IP/VPN/EVPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46, End.DX2, End.DX2V, End.DT2U and End.DT2M SIDs can be signaled in BGP.

In some scenarios an egress PE advertising a VPN route might wish to abstract the specific behavior bound to the SID from the ingress PE and other routers in the network. In such case, the SID may be advertised using the Opaque SRv6 Endpoint Behavior codepoint defined in Table 4. The details of such control plane signaling mechanisms are out of the scope of this document.

8.4. Summary

The following table summarizes behaviors for SIDs that can be signaled in which each respective control plane protocol.

| | IGP | BGP-LS | BGP IP/VPN/EVPN |
|-----------------------|-----|--------|-----------------|
| End (PSP, USP, USD) | X | X | |
| End.X (PSP, USP, USD) | X | X | |
| End.T (PSP, USP, USD) | X | X | |
| End.DX6 | X | X | X |
| End.DX4 | X | X | X |
| End.DT6 | X | X | X |
| End.DT4 | X | X | X |
| End.DT46 | X | X | X |
| End.DX2 | | X | X |
| End.DX2V | | X | X |
| End.DT2U | | X | X |
| End.DT2M | | X | X |
| End.B6.Encaps | | X | |
| End.B6.Encaps.Red | | X | |
| End.B6.BM | | X | |

Table 1: SRv6 locally instantiated SIDs signaling

The following table summarizes which SR Policy Headend capabilities are signaled in which signaling protocol.

| | IGP | BGP-LS | BGP IP/VPN/EVPN |
|-----------------|-----|--------|-----------------|
| H.Encaps | X | X | |
| H.Encaps.Red | X | X | |
| H.Encaps.L2 | | X | |
| H.Encaps.L2.Red | | X | |

Table 2: SRv6 Policy Headend behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SR policies.

For example, a BGP-LS advertisement of H.Encaps behavior would describe the capability of node N to perform a H.Encaps behavior. Specifically, it would describe how many SIDs could be pushed by N without significant performance degradation.

As a reminder, an SR policy is always assigned a Binding SID [RFC8402]. BSIDs are also advertised in BGP-LS as shown in Table 1.

Hence, the Table 2 only focuses on the generic capabilities related to H.Encaps.

9. Security Considerations

The security considerations for Segment Routing are discussed in [RFC8402]. Section 5 of [RFC8754] describes the SR Deployment Model and the requirements for securing the SR Domain. The security considerations of [RFC8754] also cover topics such as attack vectors and their mitigation mechanisms that also apply the behaviors introduced in this document. Together, they describe the required security mechanisms that allow establishment of an SR domain of trust. Having such a well-defined trust boundary is necessary in order to operate SRv6-based services for internal traffic while preventing any external traffic from accessing or exploiting the SRv6-based services. Care and rigor in IPv6 address allocation for use for SRv6 SID allocations and network infrastructure addresses, as distinct from IPv6 addresses allocated for end-users/systems (as illustrated in Section 5.1 of [RFC8754]), can provide the clear distinction between internal and external address space that is required to maintain the integrity and security of the SRv6 Domain. Additionally, [RFC8754] defines an HMAC TLV permitting SR Endpoint Nodes in the SR domain to verify that the SRH applied to a packet was selected by an authorized party and to ensure that the segment list is not modified after generation, regardless of the number of segments in the segment list. When enabled by local configuration, HMAC processing occurs at the beginning of SRH processing as defined in [RFC8754] Section 2.1.2.1 .

This document introduces SRv6 Endpoint and SR Policy Headend behaviors for implementation on SRv6 capable nodes in the network. The headend policy definition should be consistent with the specific behavior used and any local configuration (as specified in Section 4.1.1). As such, this document does not introduce any new security considerations.

The SID Behaviors specified in this document have the same HMAC TLV handling and mutability properties of the Flags, Tag, and Segment List field as the SID Behavior specified in [RFC8754].

10. IANA Considerations

10.1. Ethernet Next Header Type

This document requests IANA to allocate, in the "Protocol Numbers" registry (<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>), a new value for "Ethernet" with the following definition: The value 143 in the Next Header field of an IPv6 header

or any extension header indicates that the payload is an Ethernet frame [IEEE.802.3_2018].

IANA has done a temporary allocation of Protocol Number 143.

10.2. SRv6 Endpoint Behaviors Registry

This document requests IANA to create a new top-level registry called "Segment Routing Parameters". This registry is being defined to serve as a top-level registry for keeping all other Segment Routing sub-registries.

Additionally, a new sub-registry "SRv6 Endpoint Behaviors" is to be created under top-level "Segment Routing Parameters" registry. This sub-registry maintains 16-bit identifiers for the SRv6 Endpoint behaviors. This registry is established to provide consistency for control plane protocols which need to refer to these behaviors. These values are not encoded in the function bits within a SID.

The range of the registry is 0-65535 (0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

| Range | Hex | Registration procedure | Notes |
|-------------|----------------|-----------------------------------|---------------------|
| 0 | 0x0000 | Reserved | Not to be allocated |
| 1-32767 | 0x0001-0x7FFF | First Come First Served [RFC8126] | |
| 32768-34815 | 0x8000-0x87FF | Private Use [RFC8126] | |
| 34816-65534 | 0x8800-0xFFFFE | Reserved | Opaque |
| 65535 | 0xFFFF | Reserved | |

Table 3: SRv6 Endpoint Behaviors Registry

10.2.1. Initial Registrations

The initial registrations for the sub-registry are as follows:

| Value | Hex | Endpoint behavior | Reference |
|-------|--------|-------------------|-------------------------------|
| 0 | 0x0000 | Reserved | Not to be allocated [This.ID] |
| 1 | 0x0001 | End | |

| | | | |
|-------------|--------|----------------------------|------------------------|
| 2 | 0x0002 | End with PSP | [This.ID] |
| 3 | 0x0003 | End with USP | [This.ID] |
| 4 | 0x0004 | End with PSP&USP | [This.ID] |
| 5 | 0x0005 | End.X | [This.ID] |
| 6 | 0x0006 | End.X with PSP | [This.ID] |
| 7 | 0x0007 | End.X with USP | [This.ID] |
| 8 | 0x0008 | End.X with PSP&USP | [This.ID] |
| 9 | 0x0009 | End.T | [This.ID] |
| 10 | 0x000A | End.T with PSP | [This.ID] |
| 11 | 0x000B | End.T with USP | [This.ID] |
| 12 | 0x000C | End.T with PSP&USP | [This.ID] |
| 14 | 0x000E | End.B6.Encaps | [This.ID] |
| 15 | 0x000F | End.BM | [This.ID] |
| 16 | 0x0010 | End.DX6 | [This.ID] |
| 17 | 0x0011 | End.DX4 | [This.ID] |
| 18 | 0x0012 | End.DT6 | [This.ID] |
| 19 | 0x0013 | End.DT4 | [This.ID] |
| 20 | 0x0014 | End.DT46 | [This.ID] |
| 21 | 0x0015 | End.DX2 | [This.ID] |
| 22 | 0x0016 | End.DX2V | [This.ID] |
| 23 | 0x0017 | End.DT2U | [This.ID] |
| 24 | 0x0018 | End.DT2M | [This.ID] |
| 25 | 0x0019 | Reserved | [This.ID] |
| 27 | 0x001B | End.B6.Encaps.Red | [This.ID] |
| 28 | 0x001C | End with USD | [This.ID] |
| 29 | 0x001D | End with PSP&USD | [This.ID] |
| 30 | 0x001E | End with USP&USD | [This.ID] |
| 31 | 0x001F | End with PSP, USP & USD | [This.ID] |
| 32 | 0x0020 | End.X with USD | [This.ID] |
| 33 | 0x0021 | End.X with PSP&USD | [This.ID] |
| 34 | 0x0022 | End.X with USP&USD | [This.ID] |
| 35 | 0x0023 | End.X with PSP, USP & USD | [This.ID] |
| 36 | 0x0024 | End.T with USD | [This.ID] |
| 37 | 0x0025 | End.T with PSP&USD | [This.ID] |
| 38 | 0x0026 | End.T with USP&USD | [This.ID] |
| 39 | 0x0027 | End.T with PSP, USP & USD | [This.ID] |
| 40-32766 | | Unassigned | |
| 32767 | 0x7FFF | The SID defined in RFC8754 | [This.ID] [RFC8754] |
| 32768-65534 | | Reserved | |
| 65535 | 0xFFFF | Opaque | [This.ID] |

Table 4: IETF - SRv6 Endpoint Behaviors

11. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Thierry Couture, Kris Michielsen, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulou, Satish Damodaran, John Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya, Saleem Hafeez and Brian Carpenter.

12. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Lapishills Consulting Limited
Cyprus

Email: dirk@lapishills.com

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Google
United States of America

Email: helmalky@google.com

Prem Jonnalagadda
Barefoot Networks
United States of America

Email: prem@barefootnetworks.com

Milad Sharif
SambaNova Systems
United States of America

Email: milad.sharif@sambanova.ai

David Lebrun
Google
Belgium

Email: dlebrun@google.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Arthi Ayyangar
Arrcus, Inc
United States of America

Email: arthi@arrcus.com

Satish Mynam
Arrcus, Inc
United States of America

Email: satishm@arrcus.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma
Juniper
Singapore

Email: mashao@juniper.net

Ahmed Bashandy
Individual
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Patrice Brissette
Cisco Systems, Inc.
Canada

Email: pbrisset@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America

Email: zali@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.
India

Email: ketant@cisco.com

13. References

13.1. Normative References

- [IEEE.802.3_2018]
IEEE, "802.3-2018", IEEE 802.3-2018,
DOI 10.1109/IEEESTD.2018.8457469, August 2018,
<<https://ieeexplore.ieee.org/document/8457469>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in
IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,
December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
"IPv6 Flow Label Specification", RFC 6437,
DOI 10.17487/RFC6437, November 2011,
<<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6
(IPv6) Specification", STD 86, RFC 8200,
DOI 10.17487/RFC8200, July 2017,
<<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J.,
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header
(SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020,
<<https://www.rfc-editor.org/info/rfc8754>>.

13.2. Informative References

- [I-D.filsfils-spring-srv6-net-pgm-illustration]
Filsfils, C., Camarillo, P., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", draft-filsfils-spring-srv6-net-pgm-illustration-03 (work in progress), September 2020.
- [I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-05 (work in progress), November 2020.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <<https://www.rfc-editor.org/info/rfc4664>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <<https://www.rfc-editor.org/info/rfc8214>>.
- [RFC8317] Sajassi, A., Ed., Salam, S., Drake, J., Uttaro, J., Boutros, S., and J. Rabadan, "Ethernet-Tree (E-Tree) Support in Ethernet VPN (EVPN) and Provider Backbone Bridging EVPN (PBB-EVPN)", RFC 8317, DOI 10.17487/RFC8317, January 2018, <<https://www.rfc-editor.org/info/rfc8317>>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

John Leddy
Individual Contributor
United States of America

Email: john@leddy.net

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Satoru Matsushima
SoftBank
1-9-1, Higashi-Shimbashi, Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

C. Li
Huawei Technologies
W. Cheng
China Mobile
M. Chen
D. Dhody
Z. Li
J. Dong
Huawei Technologies
R. Gandhi
Cisco Systems, Inc.
November 4, 2019

Path Segment for SRv6 (Segment Routing in IPv6)
draft-li-spring-srv6-path-segment-04

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths by encoding paths as sequences of sub-paths, called "segments". Segment routing architecture can be implemented over an MPLS data plane as well as an IPv6 data plane.

Further, Path Segment has been defined in order to identify an SR path in SR-MPLS networks, and used for various use-cases such as end-to-end SR Path Protection and Performance Measurement (PM) of an SR path. Similar to SR-MPLS, this document defines the Path Segment in SRv6 networks in order to identify an SRv6 path.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|---|
| 1. Introduction | 2 |
| 1.1. Requirements Language | 3 |
| 1.2. Terminology | 4 |
| 2. Use Cases of SRv6 Path Segment | 4 |
| 3. SRv6 Path Segment | 5 |
| 4. SRv6 Path Segment Allocation | 5 |
| 5. Operations | 6 |
| 6. IANA Considerations | 6 |
| 7. Security Considerations | 6 |
| 8. Acknowledgements | 6 |
| 9. References | 6 |
| 9.1. Normative References | 6 |
| 9.2. Informative References | 7 |
| Authors' Addresses | 8 |

1. Introduction

Segment routing (SR) [RFC8402] is a source routing paradigm that explicitly indicates the forwarding path for packets at the ingress node by inserting an ordered list of instructions, called segments.

When segment routing is deployed on MPLS dataplane, called SR-MPLS [I-D.ietf-spring-segment-routing-mpls], a segment is an MPLS label. When segment routing is deployed on IPv6 dataplane, called SRv6 [I-D.ietf-6man-segment-routing-header], a segment is a 128 bit value, and it can be an IPv6 address of a local interface but it does not have to. For supporting SR, an extended header called Segment Routing Header (SRH), which contains a list of SIDs and several needed information such as Segments Left, has been defined in [I-D.ietf-6man-segment-routing-header].

In an SR-MPLS network, when a packet is transmitted along an SR path, the labels in the MPLS label stack will be swapped or popped, so no label or only the last label may be left in the MPLS label stack when the packet reaches the egress node. Thus, the egress node can not determine from which ingress node or SR path the packet came in. For identifying an SR-MPLS path, Path Segment is defined in [I-D.ietf-spring-mpls-path-segment].

Likewise, a path needs to be identified in an SRv6 network for several use cases such as binding bidirectional paths [I-D.li-pce-sr-bidir-path] and end-to-end performance measurement [I-D.gandhi-spring-udp-pm]. An SRv6 path can be identified by the content of segment list (i.e., the several SRv6 segments that are in the segment list).

However, the segment list may not be a good key to identify an SRv6 path, since the the length of segment list is flexible according to the number of SIDs. Also, the length of SID list will be too long to be a key when it contains many SIDs. For instance, if packet A uses the SRH with 3 SIDs while Packet B uses the SRH with 10 SIDs, the key to identify these two paths will be a 384-bits value and a 1280-bits value.

This document defines a new SRv6 segment called "SRv6 Path Segment", which is a 128-bits value, to identify an SRv6 path. Using the Path Segment as an SRv6 SID will improve performance and operations in both SR-MPLS and SRv6.

Also, In reduced mode [I-D.ietf-6man-segment-routing-header], an SRv6 path can not be indentified by the information carried by SRH. When the SRv6 Path Segment is used in reduced SRH [I-D.ietf-6man-segment-routing-header], the entire path information is indicated by the Path Segment, and the performance will be better than using SID list as the path identifier, while the overhead equals to the normal SRH.

The SRv6 Path Segment will be used for identifying an SRv6 path and path related services, and it will not be updated to the IPv6 destination address, so it is not routable.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

MPLS: Multiprotocol Label Switching.

PM: Performance Measurement.

SID: Segment ID.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

SRH: Segment Routing Header.

PSID: Path Segment Identifier.

PSP: Penultimate Segment Popping.

Further, this document makes use of the terms defined in [RFC8402] and [I-D.ietf-spring-srv6-network-programming].

2. Use Cases of SRv6 Path Segment

Similar to SR-MPLS Path Segment [I-D.ietf-spring-mpls-path-segment], SRv6 Path Segment also can be used for identifying an SRv6 Path in some use cases:

- o Performance Measurement: For Passive measurement [RFC7799], path identification at the measuring points is the pre-requisite [I-D.ietf-spring-mpls-path-segment]. SRv6 Path segment can be used by the measuring points (e.g., the ingress/egress nodes of an SRv6 path) or a centralized controller to correlate the packets counts/timestamps, then packet loss/delay can be calculated.
- o Bi-directionioinal SRv6 Path Association: In some scenarios, such as mobile backhaul transport network, there are requirements to support bidirectional path. Similar to SR-MPLS [I-D.ietf-spring-mpls-path-segment], to support bidirectional SRv6 path, a straightforward way is to bind two unidirectional SRv6 paths to a single bidirectional path. SRv6 Path segments can be used to correlate the two unidirectional SRv6 paths at both ends of the paths. [I-D.li-pce-sr-bidir-path] defines how to use PCEP and Path segment to initiate a bidirectional SR path.
- o End-to-end Path Protection: For end-to-end 1+1 path protection (i.e., Live-Live case), the egress node of an SRv6 path needs to know the set of paths that constitute the primary and the secondary(s), in order to select the primary packet for onward

transmission, and to discard the packets from the secondary(s), so each SRv6 path needs a unique path identifier at the egress node, which can be an SRv6 Path Segment.

3. SRv6 Path Segment

As defined in [I-D.ietf-spring-srv6-network-programming], an SRv6 segment is a 128-bit value.

In order to identify an SRv6 path, this document defines a new segment called SRv6 Path Segment.

The SRv6 Path Segment MUST appear only once in a SID list. The detailed encoding of SRv6 Path Segment is out of scope of this document, and it is defined in [I-D.li-6man-srv6-path-segment-encap].

Depending on the use case, an SRv6 Path Segment identifies:

- o an SRv6 path within an SRv6 domain
- o an SRv6 Policy
- o a Candidate-paths or a SID-List in a SRv6 Policy
[I-D.ietf-spring-segment-routing-policy]

Note that, based on the use-case, the different SID-Lists of SR Policies may use the same SRv6 Path Segment.

4. SRv6 Path Segment Allocation

A Path Segment is a local segment allocated by an egress node. A Path Segment can be allocated through several ways, such as CLI, BGP [I-D.ietf-idr-sr-policy-path-segment], PCEP [I-D.ietf-pce-sr-path-segment] or other ways. The mechanisms through which a Path Segment is allocated is out of scope of this document.

When the Path Segment is allocated by the egress, it MUST be distributed to the ingress node. In this case, only the egress will process the Path Segment, and other nodes specified by SIDs in the SID list do not know how to process the Path Segment.

Depending on the use case, a Path Segment may be distributed to the SRv6 nodes along the SRv6 path. In this case, the SRv6 nodes that learned Path Segment may process the Path Segment depending on the use case.

5. Operations

An egress node or other SRv6 nodes along the SRv6 path supporting the Path Segment processing will inspect the last entry of the segment list (giving the the node will inspect the last entry in the SID list and obtain the Path Segment. The processing of the Path Segment is described in [I-D.li-6man-srv6-path-segment-encap].

6. IANA Considerations

This document does not require any IANA actions.

7. Security Considerations

This document does not introduce additional security requirements and mechanisms other than the ones described in [RFC8402].

8. Acknowledgements

The authors would like to thank Stefano Previdi and Zafar Ali for their valuable comments and suggestions.

9. References

9.1. Normative References

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-05 (work in progress), October 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

9.2. Informative References

- [I-D.gandhi-spring-udp-pm]
Gandhi, R., Filsfils, C., daniel.voyer@bell.ca, d., Salsano, S., Ventre, P., and M. Chen, "UDP Path for In-band Performance Measurement for Segment Routing Networks", draft-gandhi-spring-udp-pm-02 (work in progress), September 2018.
- [I-D.ietf-idr-sr-policy-path-segment]
Li, C., Li, Z., Telecom, C., Cheng, W., and K. Talaulikar, "SR Policy Extensions for Path Segment and Bidirectional Path", draft-ietf-idr-sr-policy-path-segment-00 (work in progress), October 2019.
- [I-D.ietf-pce-sr-path-segment]
Li, C., Chen, M., Cheng, W., Gandhi, R., and Q. Xiong, "Path Computation Element Communication Protocol (PCEP) Extension for Path Segment in Segment Routing (SR)", draft-ietf-pce-sr-path-segment-00 (work in progress), October 2019.
- [I-D.ietf-spring-mpls-path-segment]
Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler, "Path Segment in MPLS Based Segment Routing Network", draft-ietf-spring-mpls-path-segment-01 (work in progress), September 2019.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.

[I-D.li-6man-srv6-path-segment-encap]

Li, C., Cheng, W., Li, Z., and D. Dhody, "Encapsulation of Path Segment in SRv6", draft-li-6man-srv6-path-segment-encap-00 (work in progress), July 2019.

[I-D.li-pce-sr-bidir-path]

Li, C., Chen, M., Cheng, W., Li, Z., Dong, J., Gandhi, R., and Q. Xiong, "PCEP Extensions for Associated Bidirectional Segment Routing (SR) Paths", draft-li-pce-sr-bidir-path-06 (work in progress), August 2019.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.

Authors' Addresses

Cheng Li
Huawei Technologies

Email: chengli13@huawei.com

Weiqiang Cheng
China Mobile

Email: chengweiqiang@chinamobile.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

Email: dhruv.ietf@gmail.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Rakesh Gandhi
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2021

C. Li
Huawei Technologies
W. Cheng
China Mobile
M. Chen
D. Dhody
Huawei Technologies
R. Gandhi
Cisco Systems, Inc.
October 31, 2020

Path Segment for SRv6 (Segment Routing in IPv6)
draft-li-spring-srv6-path-segment-07

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths by encoding an ordered list of instructions, called "segments". The SR architecture can be implemented over an MPLS data plane as well as an IPv6 data plane.

Currently, Path Segment has been defined to identify an SR path in SR-MPLS networks, and is used for various use-cases such as end-to-end SR Path Protection and Performance Measurement (PM) of an SR path. This document defines the Path Segment to identify an SRv6 path in an IPv6 network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 4, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Requirements Language | 4 |
| 1.2. Terminology | 4 |
| 2. Use Cases for SRv6 Path Segment | 4 |
| 3. SRv6 Path Segment | 5 |
| 3.1. Format of an SRv6 Path Segment | 5 |
| 3.1.1. SRv6 Path Segment: Locator and Local ID | 5 |
| 3.1.2. SRv6 Path Segment: Global ID | 6 |
| 4. SRv6 Path Segment Allocation | 6 |
| 5. Processing of SRv6 Path Segment | 7 |
| 6. IANA Considerations | 7 |
| 7. Security Considerations | 8 |
| 8. Contributors | 8 |
| 9. Acknowledgements | 8 |
| 10. References | 8 |
| 10.1. Normative References | 8 |
| 10.2. Informative References | 9 |
| Authors' Addresses | 10 |

1. Introduction

Segment routing (SR) [RFC8402] is a source routing paradigm that explicitly indicates the forwarding path for packets at the ingress node by inserting an ordered list of instructions, called segments.

When segment routing is deployed on an MPLS data plane, called SR-MPLS [I-D.ietf-spring-segment-routing-mpls], a segment identifier (SID) is present as an MPLS label. When segment routing is deployed on an IPv6 data plane, a SID is presented as a 128-bit value, and it can be an IPv6 address of a local interface but it does not have to

be. To support SR in an IPv6 network, a Segment Routing Header (SRH) [RFC8754] is used.

In an SR-MPLS network, when a packet is transmitted along an SR path, the labels in the MPLS label stack will be swapped or popped, so no label or only the last label may be left in the MPLS label stack when the packet reaches the egress node. Thus, the egress node can not determine from which ingress node or SR path the packet came from. Therefore, to identify an SR-MPLS path, a Path Segment is defined in [I-D.ietf-spring-mpls-path-segment].

Likewise, a path needs to be identified in an SRv6 network for several use cases such as binding bidirectional paths [I-D.ietf-pce-sr-bidir-path] and end-to-end performance measurement [I-D.gandhi-spring-udp-pm].

An SRv6 path MAY be identified by the content of a segment list. However, the segment list may not be a good key, since the length of a segment list is flexible according to the number of required SIDs. Also, the length of a segment list may be too long to be a key when it contains many SIDs. For instance, if packet A uses an SRH with 3 SIDs while Packet B uses an SRH with 10 SIDs, the key to identify these two paths will be a 384-bits value and a 1280-bits value, respectively. Further, an SRv6 path cannot be identified by the information carried by the SRH in reduced mode [RFC8754] as the first SID is not present.

Furthermore, different SRv6 policies may use the same segment list for different candidate paths, so the traffic of different SRv6 policies are merged, resulting in the inability to measure the performance of the specific path.

To solve the above issues, this document defines a new SRv6 segment called "SRv6 Path Segment", which is a 128-bits value, to identify an SRv6 path.

When the SRv6 Path Segment is used in reduced mode SRH [RFC8754], the entire path information is indicated by the Path Segment, and the performance will be better than using the entire segment list as the path identifier, while the overhead is equivalent to the SRH in normal mode. Furthermore, with SRv6 Path Segment, each SRv6 candidate path can be identified and measured, even when they use the same segment list.

An SRv6 Path Segment MUST NOT be copied to the IPv6 destination address, so it is not routable.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

MPLS: Multiprotocol Label Switching.

PM: Performance Measurement.

SID: Segment ID.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

SRH: Segment Routing Header.

PSID: Path Segment Identifier.

PSP: Penultimate Segment Popping.

Further, this document makes use of the terms defined in [RFC8402] and [I-D.ietf-spring-srv6-network-programming].

2. Use Cases for SRv6 Path Segment

Similar to SR-MPLS Path Segment [I-D.ietf-spring-mpls-path-segment], SRv6 Path Segment may also be used to identify an SRv6 Path in some use cases:

- o Performance Measurement: For Passive measurement [RFC7799], path identification at the measuring points is the pre-requisite [I-D.ietf-spring-mpls-path-segment]. SRv6 Path segment can be used by the measuring points (e.g., the ingress/egress nodes of an SRv6 path) or a centralized controller to correlate the packets counts/timestamps, then packet loss/delay can be calculated.
- o Bi-directional SRv6 Path Association: In some scenarios, such as mobile backhaul transport networks, there are requirements to support bidirectional paths. Like SR-MPLS [I-D.ietf-spring-mpls-path-segment], to support bidirectional SRv6 paths, a straightforward way is to bind two unidirectional SRv6 paths to a single bidirectional path. SRv6 Path segments can be

used to correlate the two unidirectional SRv6 paths at both ends of the path. [I-D.ietf-pce-sr-bidir-path] defines how to use PCEP and Path Segment to initiate a bidirectional SR path.

- o End-to-end Path Protection: For end-to-end 1+1 path protection (i.e., Live-Live case), the egress node of an SRv6 path needs to know the set of paths that constitute the primary and the secondary(s), to select the primary packet for onward transmission, and to discard the packets from the secondary(s), so each SRv6 path needs a unique path identifier at the egress node, which can be an SRv6 Path Segment.

3. SRv6 Path Segment

As defined in [I-D.ietf-spring-srv6-network-programming], an SRv6 segment is a 128-bit value.

To identify an SRv6 path, this document defines a new segment called SRv6 Path Segment.

The SRv6 Path Segment MUST appear only once in a segment list, and it MUST appear as the last entry in the segment list. To indicate the SRv6 Path Segment, an SRH.P-flag is defined in [I-D.li-6man-srv6-path-segment-encap].

Depending on the use case, an SRv6 Path Segment identifies:

- o an SRv6 path within an SRv6 domain
- o an SRv6 Policy
- o a Candidate-path or a SID-List in a SRv6 Policy
[I-D.ietf-spring-segment-routing-policy]

Note that, based on the use-case, a SRv6 Path Segment can be used for different SID-Lists within an SR Policy.

3.1. Format of an SRv6 Path Segment

This document proposes two types of SRv6 Path Segment format.

3.1.1. SRv6 Path Segment: Locator and Local ID

As per [I-D.ietf-spring-srv6-network-programming], an SRv6 segment is a 128-bit value, which can be represented as LOC:FUNCT, where LOC is the L most significant bits and FUNCT is the 128-L least significant bits. L is called the locator length and is flexible. Each network operator is free to use the locator length it chooses. Most often

the LOC part of the SID is routable and leads to the node which instantiates that SID. The FUNCT part of the SID is an opaque identification of a local function bound to the SID. The FUNCT value zero is invalid.

SRv6 Path Segment can follow the format, where the LOC part identifies the egress node that allocates the Path Segment, and the FUNCT part is a unique local ID to identify an SRv6 Path and its endpoint behavior.

The Function Type of an SRv6 Path Segment is END.PSID (End Function with Path Segment Identifier).

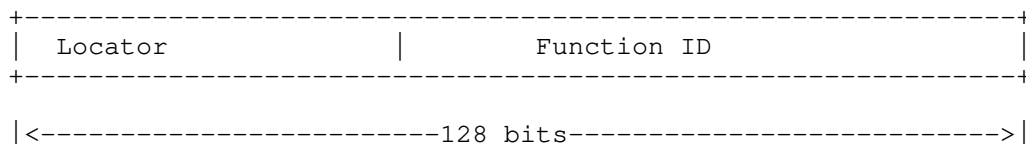


Figure 2. PSID in Format LOC:FUNCT

3.1.2. SRv6 Path Segment: Global ID

An SRv6 Path Segment ID can be a Global ID, and its format depends on the use case.

The SRv6 Path Segment will not be copied to the IPv6 Destination Address, so the SRv6 Path Segment ID can be allocated from an independent 128-bits ID Space. In this case, a new table should be maintained at the node for SRv6 Path Segment.

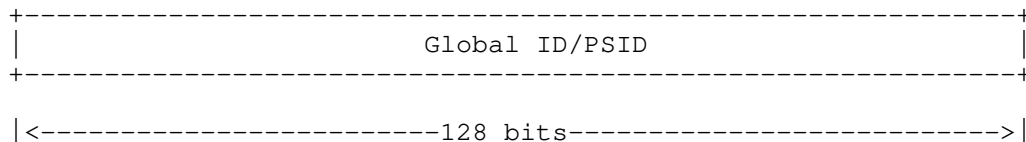


Figure 3. A Global ID as an PSID

4. SRv6 Path Segment Allocation

A Path Segment is a local segment allocated by an egress node. A Path Segment can be allocated through several ways, such as CLI, BGP [I-D.ietf-idr-sr-policy-path-segment], PCEP [I-D.ietf-pce-sr-path-segment] or other ways. The mechanisms through which a Path Segment is allocated are out of scope of this document.

When a Path Segment is allocated by the egress, it MUST be distributed to the ingress node of the path that identified by the path segment. In this case, only the egress will process the Path Segment, and other nodes specified by SIDs in the segment list do not know how to process the Path Segment.

Depending on the use case, a Path Segment may be distributed to the SRv6 nodes along the SRv6 path. In this case, the SRv6 nodes that learned the Path Segment may process the Path Segment depending on the use case.

5. Processing of SRv6 Path Segment

When the SRv6 Path Segment is used, the following rules apply:

- o The SRv6 Path Segment MUST appear only once in a segment list, and it MUST appear as the last entry. Only the one that appears as the last entry in the SID list will be processed. An SRv6 Path Segment that appears at any other location in the SID list will be treated as an error.
- o When an SRv6 Path Segment is inserted, the SL MUST be initiated to be less than the value of Last Entry, and will not point to SRv6 Path Segment. For instance, when the Last entry is 4, the SID List[4] is the SRv6 Path Segment, so the SL MUST be set to 3 or other numbers less than Last entry.
- o The SRv6 Path Segment MUST NOT be copied to the IPv6 destination address.
- o Penultimate Segment Popping (PSP, as defined in [I-D.ietf-spring-srv6-network-programming]) MUST be disabled.
- o The ingress needs to set the P-bit when an SRv6 Path Segment is inserted in the SID List. Nodes that support SRv6 Path Segment processing will inspect the last entry to process SRv6 Path Segment when the P-bit is set. When the P-bit is unset, the nodes will not inspect the last entry.
- o The specific SRv6 Path Segment processing depends on use cases, and it is out of scope of this document.

6. IANA Considerations

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Behaviors" sub-registry belonging to the top-level "Segment-routing with IPv6 data plane (SRv6) Parameters" registry, the following allocations:

| Value | Description | Reference |
|-------|------------------------------|-----------|
| TBA1 | End.PSID - SRv6 Path Segment | [This.ID] |

7. Security Considerations

This document does not introduce additional security requirements and mechanisms other than the ones described in [RFC8402].

8. Contributors

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

9. Acknowledgements

The authors would like to thank Stefano Previdi and Zafar Ali for their valuable comments and suggestions.

10. References

10.1. Normative References

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D.,
Matsushima, S., and Z. Li, "SRv6 Network Programming",
draft-ietf-spring-srv6-network-programming-24 (work in
progress), October 2020.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

10.2. Informative References

- [I-D.gandhi-spring-udp-pm]
Gandhi, R., Filsfils, C., daniel.voyer@bell.ca, d., Salsano, S., Ventre, P., and M. Chen, "UDP Path for In-band Performance Measurement for Segment Routing Networks", draft-gandhi-spring-udp-pm-02 (work in progress), September 2018.
- [I-D.ietf-idr-sr-policy-path-segment]
Li, C., Li, Z., Telecom, C., Cheng, W., and K. Talaulikar, "SR Policy Extensions for Path Segment and Bidirectional Path", draft-ietf-idr-sr-policy-path-segment-01 (work in progress), August 2020.
- [I-D.ietf-pce-sr-bidir-path]
Li, C., Chen, M., Cheng, W., Gandhi, R., and Q. Xiong, "PCEP Extensions for Associated Bidirectional Segment Routing (SR) Paths", draft-ietf-pce-sr-bidir-path-03 (work in progress), September 2020.
- [I-D.ietf-pce-sr-path-segment]
Li, C., Chen, M., Cheng, W., Gandhi, R., and Q. Xiong, "Path Computation Element Communication Protocol (PCEP) Extension for Path Segment in Segment Routing (SR)", draft-ietf-pce-sr-path-segment-01 (work in progress), May 2020.

[I-D.ietf-spring-mpls-path-segment]

Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler,
"Path Segment in MPLS Based Segment Routing Network",
draft-ietf-spring-mpls-path-segment-03 (work in progress),
September 2020.

[I-D.ietf-spring-segment-routing-mpls]

Bashandy, A., Filsfils, C., Previdi, S., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-22
(work in progress), May 2019.

[I-D.ietf-spring-segment-routing-policy]

Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and
P. Mattes, "Segment Routing Policy Architecture", draft-
ietf-spring-segment-routing-policy-08 (work in progress),
July 2020.

[I-D.li-6man-srv6-path-segment-encap]

Li, C., Cheng, W., Li, Z., and D. Dhody, "Encapsulation of
Path Segment in SRv6", draft-li-6man-srv6-path-segment-
encap-03 (work in progress), September 2020.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with
Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799,
May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.

Authors' Addresses

Cheng Li
Huawei Technologies

Email: c.l@huawei.com

Weiqliang Cheng
China Mobile

Email: chengweiqliang@chinamobile.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

Email: dhruv.ietf@gmail.com

Rakesh Gandhi
Cisco Systems, Inc.
Canada

Email: rgandhi@cisco.com

Network
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2020

C. Weiqiang
China Mobile
G. Mirsky
ZTE Corp.
P. Shaofu
L. Aihua
ZTE Corporation
W. Xiaolan
New H3C Technologies Co. Ltd
C. Wei
Centec
S. Zadok
Broadcom
November 3, 2019

Unified Identifier in IPv6 Segment Routing Networks
draft-mirsky-6man-unified-id-sr-04

Abstract

Segment Routing architecture leverages the paradigm of source routing. It can be realized in a network data plane by prepending the packet with a list of instructions, a.k.a. segments. A segment can be encoded as a Multi-Protocol Label Switching (MPLS) label, IPv4 address, or IPv6 address. Segment Routing can be applied in MPLS data plane by encoding segments in MPLS label stack. It also can be applied to IPv6 data plane by encoding a list of segment identifiers in IPv6 Segment Routing Extension Header (SRH). This document extends the use of the SRH to unified identifiers encoded as MPLS label or IPv4 address, to compress the SRH, and support support more detailed network programming and interworking between SR-MPLS and SRv6 domains.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Conventions used in this document | 3 |
| 1.1.1. Terminology | 3 |
| 1.1.2. Requirements Language | 4 |
| 2. Segment Routing Extension Header: Benefits and Challenges . . | 4 |
| 3. Unified SIDs in IPv6 Segment Routing Extension Header | 4 |
| 4. The Use Case of Unified Segment Identifier | 6 |
| 4.1. Interworking Between SR-MPLS and SRv6 Using U-SID | 6 |
| 5. Operations with Unified Segment Identifier | 7 |
| 5.1. Procedures of SR-MPLS over IP | 8 |
| 5.2. Packet Forwarding | 8 |
| 6. IANA Considerations | 10 |
| 7. Security Considerations | 10 |
| 8. Acknowledgements | 10 |
| 9. Normative References | 10 |
| Authors' Addresses | 11 |

1. Introduction

Segment Routing architecture [RFC8402] leverages the paradigm of source routing. It can be realized in a network data plane by prepending the packet with a list of instructions, a.k.a. segment identifiers (SIDs). A segment can be encoded as a Multi-Protocol Label Switching (MPLS) label, IPv4 address, or IPv6 address. Segment Routing can be applied in MPLS data plane by encoding 20-bits SIDs in MPLS label stack [I-D.ietf-spring-segment-routing-mpls]. It also can be applied to IPv6 data plane by encoding a list of 128-bits SIDs in IPv6 Segment Routing Extension Header (SRH)

[I-D.ietf-6man-segment-routing-header]. Applicability of 32-bits SID that may represent an IPv4 address has not been defined.

SR extensions to Interior Gateway Protocols (IGP), IS-IS [I-D.ietf-isis-segment-routing-extensions], OSPF [I-D.ietf-ospf-segment-routing-extensions], and OSPFv3 [I-D.ietf-ospf-ospfv3-segment-routing-extensions], defined how 20-bits and 32-bits SIDs advertised and bound to SR objects and/or instructions. Extensions to BGP link-state address family [I-D.ietf-idr-bgp-ls-segment-routing-ext] enabled propagation of segment information of variable length via BGP.

This document extends the use of the SRH [I-D.ietf-6man-segment-routing-header] to unified identifiers encoded as MPLS label or IPv4 address to support more detailed network programming and interworking between SR-MPLS and SRv6 domains.

1.1. Conventions used in this document

1.1.1. Terminology

SR: Segment Routing

SRH: Segment Routing Extension Header

MPLS: Multiprotocol Label Switching

SR-MPLS: Segment Routing using MPLS data plane

SID: Segment Identifier

IGP: Interior Gateway Protocol

DA: Destination Address

ILM: Incoming Label Map

FEC: Forwarding Equivalence Class

FTN: FEC-to-NHLFE map

OAM: Operation, Administration and Maintenance

TE: Traffic Engineering

SRv6: Segment Routing in IPv6

U-SID: Unified Segment Identifier

PSP: Penultimate Segment Popping

FIB: Forwarding Information Base

1.1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Segment Routing Extension Header: Benefits and Challenges

Many functions related to Operation, Administration and Maintenance (OAM) require identification of the SR tunnel ingress and the path, constructed by segments, between the ingress and the egress SR nodes. Combination of IPv6 encapsulation [RFC8200] and SRH [I-D.ietf-6man-segment-routing-header], referred to as SRv6, comply with these requirements while it is challenging when applying SR in MPLS networks, also referred to as SR-MPLS.

On the other hand, the size of IPv6 SID presents a scaling challenge to use topological instructions that define strict explicit traffic engineered (TE) path or support network programming in combination with service-based instructions. At the same time, that is where SR-MPLS approach provides better results due to smaller SID length. It can be used to compress the SRv6 header size when a smaller namespace of available SIDs is sufficient for addressing the particular network.

SR-MPLS is broadly used in metro networks. With the gradual deployment of SRv6 in the core networks, supporting interworking between SR-MPLS and SRv6 becomes the necessity for operators. It is operationally more efficient and straightforward if SRv6 can use the same size SIDs as in SR-MPLS. The SRH can be extended to define the same as in SR-MPLS SID length to support the unified segment identifier (U-SID). As a result, end-to-end SR tunnel may use U-SIDs across SR-MPLS and SRv6 domains.

3. Unified SIDs in IPv6 Segment Routing Extension Header

SRH format has been defined in Section 3 of [I-D.ietf-6man-segment-routing-header] as presented in Figure 1

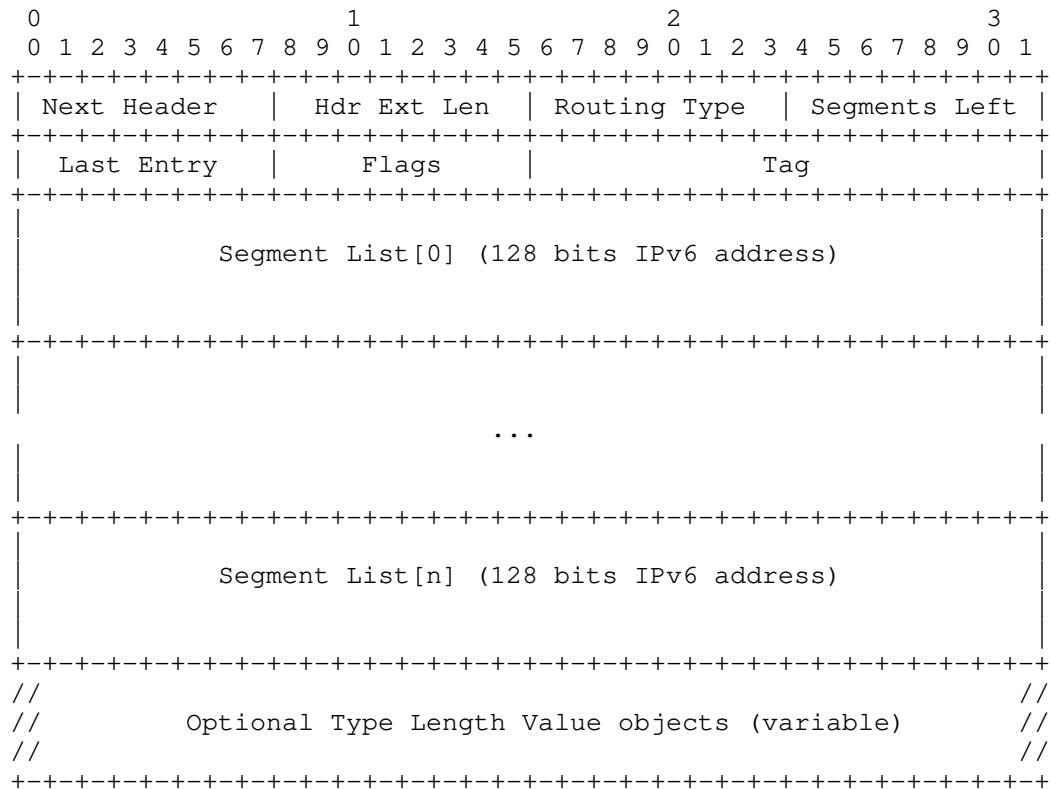


Figure 1: SRH format

This document defines a new field Size in the SRH Flags field as a two-bits field with the following values:

0b00 - 128-bits SID, an IPv6 address;

0b01 - 32-bits SID, an IPv4 address;

0b10 - 32-bits SID, an MPLS label in leftmost 20-bits, rightmost 12-bits for context information used by the label forwarding entry. The context information could be U-SID function code.

0b11 - reserved for future use.

Entries of the segment list in the SRH MUST be of the same length.

4. The Use Case of Unified Segment Identifier

U-SID can be used for interworking between SR-MPLS and SRv6 domains. SR-MPLS is often used in a metro network, for example, in the backhaul metro network of CMCC. If the core network uses SRv6, for example, the core network of the same operator, U-SID can be used in the SRv6 domain to interwork with SR-MPLS in the metro network to form an end-to-end tunnel.

4.1. Interworking Between SR-MPLS and SRv6 Using U-SID

SR-MPLS uses SR SIDs as MPLS label in MPLS stack, and the SIDs are 32-bits long. SRv6 uses SR SIDs as IPv6 extension header in SRH, and the SIDs are 128-bits long.

The U-SID uses the same 32-bits long SIDs in MPLS stack and SRH. Thus, four 32-bits long U-SIDs can be placed in the space of a single 128-bits long header. The encapsulation is illustrated in Figure 2.

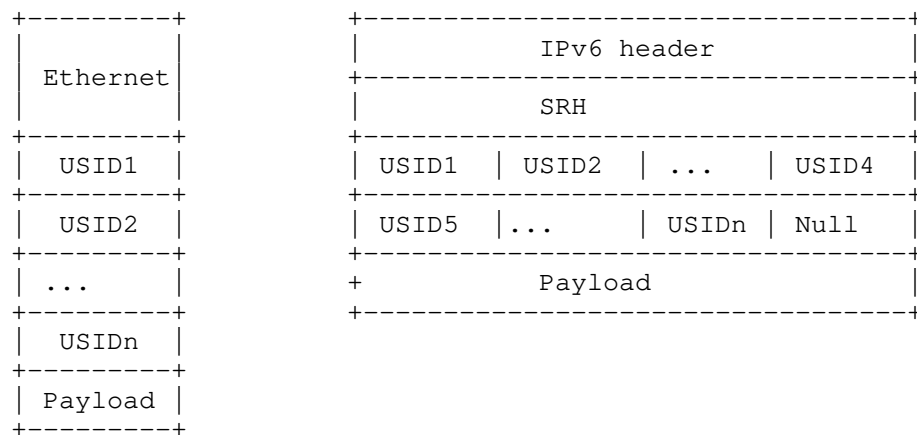


Figure 2: 32-bits long U-SIDs Encapsulation

The SR-MPLS and SRv6 interworking is illustrated in Figure 3. An end-to-end SR tunnel from A to F crosses the SR-MPLS and SRv6 domains. The SR-MPLS domain could be using IPv4 or IPv6 address family. The SRv6 border nodes (E/G) receive SR-MPLS packets and forward them into the SRv6 domain using an SR-MPLS Binding SID [I-D.ietf-spring-segment-routing-mpls].

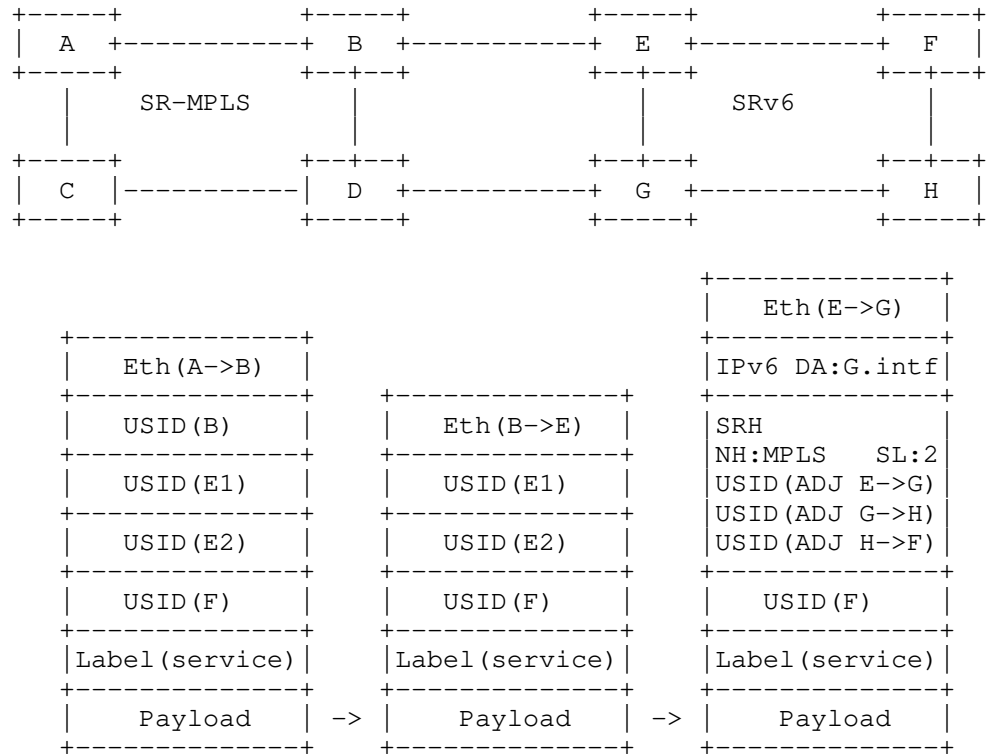


Figure 3: SR-MPLS and SRv6 interworking

The SRv6 edge node E assigns two SIDs, e.g., E1 and E2, E1 is an SR-MPLS Node-SID, E2 is an SR-MPLS Binding-SID, which represents an SRv6 policy (from E to F, via segment list E-G-H-F) with U-SID encapsulation. Figure 4 demonstrates an example of the packet forwarding, where U-SID is an MPLS label.

The controller may assign the end-to-end SR tunnel U-SIDs (from A to F), and another method is outside the scope of this document.

5. Operations with Unified Segment Identifier

When SRH is used to include 32-bits long U-SIDs, the ingress and transit nodes of an SR tunnel act as described in Section 5.1 and Section 5.2 of [I-D.ietf-6man-segment-routing-header] respectively.

If U-SID is used to support interworking between SR-MPLS and SRv6 domains, it is beneficial that U-SID type matches to an MPLS label. In that case, an ILM (Incoming Label Map) entry can be used to map a

U-SID to an IPv6 address. As result, it is not necessary to introduce a new type of index-based mapping table. For ILM entry of Adjacency-SID, the mapping result copied to DA (Destination Address) is the remote interface IPv6 address, for ILM entry of Node-SID, the mapping result copied to DA is remote node loopback IPv6 address.

Operations on an MPLS label of U-SID type are the same as those defined in [I-D.ietf-mpls-sr-over-ip]. However, SR-MPLS over SRH has the following advantages compared with SR-MPLS over UDP:

- o SRH is flexible to extend flags or sub-TLVs for service requirements, but UDP not.
- o Labels in SRH can meet 8 bytes alignment requirements as per [RFC8200], but UDP not.
- o The source address of the SR policy is not discarded, but UDP not.

5.1. Procedures of SR-MPLS over IP

Procedures of SR-MPLS over IP of [I-D.ietf-mpls-sr-over-ip] described how to construct an adjusted SR-MPLS FTN (FEC-to-NHLFE map) and ILM entry towards a prefix-SID when next-hops are IP-only routers, the action of FTN and ILM entry will steer the packet along an outer tunnel to the target node that originated the FEC (Forwarding Equivalence Class), and on each airway node along the segment list, UDP header is frequently removed and put again. However, for SR-MPLS over SRH in this document we don't try to depend on that adjusted FIB (Forwarding Information Base) entry, because there are not any actions needed to get from the FIB entry, a traditional ILM entry (maybe without out-label because of IP-only next-hop) is enough to get the FEC information, i.e., to map a U-SID to an IPv6 address and copy to DA. An SRv6 policy chosen to encapsulate U-SID list within SRH is determined at the ingress node of this SRv6 policy, SRH is preserved along the SR to egress, though PSP (Penultimate Segment Popping) may be used, that is different from SR-MPLS over IP/UDP method [I-D.ietf-mpls-sr-over-ip], so the source address (i.e., the ingress of the SRv6 policy) is not discarded.

5.2. Packet Forwarding

U-SID based packet forwarding is similar to the processing described in [I-D.ietf-mpls-sr-over-ip]. But it differs from that in FIB action and segment list processing. For completeness, we repeat the description of [I-D.ietf-mpls-sr-over-ip] with modification as follows.

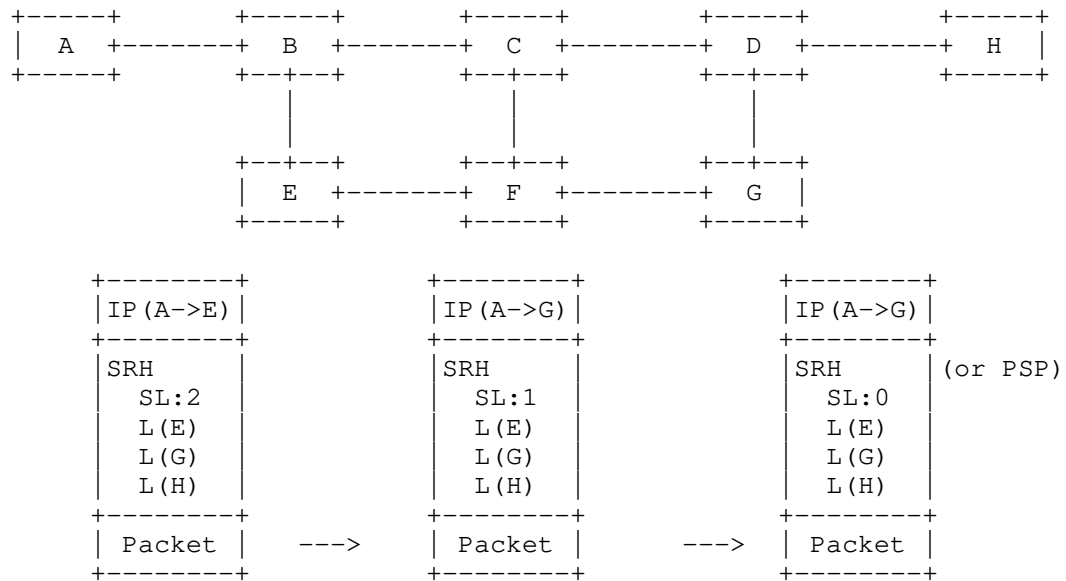


Figure 4: Packet Forwarding Example

In the example shown in Figure 4, assume that routers A, E, G, and H are U-SID capable (i.e, both SR-MPLS and SRv6 capable) while the remaining routers (B, C, D, and F) are only capable of forwarding IP packets. Routers A, E, G, and H advertise their Segment Routing related information via IS-IS or OSPF.

Now assume that router A (the Domain ingress) wants to send a packet to router H (the Domain egress) via an SRv6 policy with the explicit path {E->G->H}. Router A will impose an MPLS label stack within SRH on the packet that corresponds to that explicit path. Router A searches ILM entry by the top label (that indicated router E), get the FEC information, a loopback IPv6 address of E, and then copy to DA and sends the packet. The value of SRH.SL is 2.

When the IPv6 packet arrives at router E, router E get the next segment (label) within SRH according to SL 2, searches ILM entry by the next label, get the FEC information, a loopback IPv6 address of G, and then copy to DA and sends the packet. The value of SRH.SL is 1.

When the IPv6 packet arrives at router G, router G gets the next segment (label) within SRH according to SRH.SL 1, looks up ILM entry by the next label, gets the FEC information, a loopback IPv6 address of H, and then copies it to IP DA and transmits the packet. Because

the value of SRH.SL is 0, the SRH can be removed if the Prefix-SID of H is set to PSP.

6. IANA Considerations

IANA is requested to allocate from the Segment Routing Header Flags registry the two-bits long field referred to as Size.

7. Security Considerations

This specification inherits all security considerations of [RFC8402] and [I-D.ietf-6man-segment-routing-header].

8. Acknowledgements

TBD

9. Normative References

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.
- [I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-16 (work in progress), June 2019.
- [I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-25 (work in progress), May 2019.
- [I-D.ietf-mpls-sr-over-ip]
Xu, X., Bryant, S., Farrel, A., Hassan, S., Henderickx, W., and Z. Li, "SR-MPLS over IP", draft-ietf-mpls-sr-over-ip-07 (work in progress), June 2019.
- [I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P. and S. Previdi, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-23 (work in progress), January 2019.

- [I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,
Shakir, R., Henderickx, W., and J. Tantsura, "OSPF
Extensions for Segment Routing", draft-ietf-ospf-segment-
routing-extensions-27 (work in progress), December 2018.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-22
(work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6
(IPv6) Specification", STD 86, RFC 8200,
DOI 10.17487/RFC8200, July 2017,
<<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Cheng Weiqiang
China Mobile
Beijing
China

Email: chengweiqiang@chinamobile.com

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Peng Shaofu
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: peng.shaofu@zte.com.cn

Liu Aihua
ZTE Corporation
Zhongxing Industrial Park, Nanshan District
Shenzhen
China

Email: liu.aihua@zte.com.cn

Wan Xiaolan
New H3C Technologies Co. Ltd
No.8, Yongjia Road, Haidian District
Beijing
China

Email: wxlan@h3c.com

Cheng Wei
Centec
Building B, No.5 Xing Han Street, Suzhou Industrial Park
Suzhou
China

Email: Chengw@centecnetworks.com

Shay
Broadcom
Israel

Email: shay.zadok@broadcom.com

Network
Internet-Draft
Intended status: Standards Track
Expires: 29 March 2022

C. Weiqiang
China Mobile
G. Mirsky
Ericsson
P. Shaofu
L. Aihua
ZTE Corporation
G. Mishra
Verizon Inc.
25 September 2021

Unified Identifier in IPv6 Segment Routing Networks
draft-mirsky-6man-unified-id-sr-10

Abstract

Segment Routing architecture leverages the paradigm of source routing. It can be realized in a network data plane by prepending the packet with a list of instructions, a.k.a. segments. A segment can be encoded as a Multi-Protocol Label Switching (MPLS) label, IPv4 address, or IPv6 address. Segment Routing can be applied in the MPLS data plane by encoding segments in the MPLS label stack. It also can be applied to the IPv6 data plane by encoding a list of segment identifiers in IPv6 Segment Routing Extension Header (SRH). This document extends the use of the SRH to unified segment identifiers encoded, for example, as MPLS label or IPv4 address, to compress the SRH, and support more detailed network programming and interworking between SR-MPLS and SRv6 domains.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 29 March 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 1.1. Conventions used in this document | 3 |
| 1.1.1. Acronyms | 3 |
| 1.1.2. Requirements Language | 4 |
| 2. Segment Routing Extension Header: Benefits and Challenges . . | 4 |
| 3. Unified SIDs in IPv6 Segment Routing Extension Header | 4 |
| 4. Operations with Unified Segment Identifier | 6 |
| 4.1. Procedures of 32bits MPLS Label within SRH | 7 |
| 4.1.1. Packet Forwarding Based on UET-MPLS U-SID | 8 |
| 4.2. Procedures of 32bits IP Address within SRH | 9 |
| 4.2.1. Packet Forwarding Based on UET-32 U-SID | 10 |
| 5. The Use Case of Unified Segment Identifier | 11 |
| 5.1. Nesting Interworking Between SR-MPLS and SRv6 Using Binding U-SID | 12 |
| 5.2. Flat Interworking Between Different UET Domains Using Mixing U-SID | 15 |
| 5.2.1. UET Capability Advertisement | 15 |
| 5.2.2. SRv6 SID Allocated per UEC | 16 |
| 5.2.3. Packets Forwarding Procedures | 17 |
| 6. Control Plane in Support of Unified SID | 21 |
| 7. SRH with U-SID Pseudo-code | 22 |
| 8. U-SID supporting SRv6 programming | 23 |
| 9. Benefits | 24 |
| 10. Implementation Considerations | 24 |
| 11. IANA Considerations | 24 |
| 12. Security Considerations | 24 |
| 13. Contributors | 24 |
| 14. Acknowledgements | 25 |
| 15. Normative References | 25 |
| Authors' Addresses | 27 |

1. Introduction

Segment Routing architecture [RFC8402] leverages the paradigm of source routing. It can be realized in a network data plane by prepending the packet with a list of instructions, a.k.a. segment identifiers (SIDs). A segment can be encoded as a Multi-Protocol Label Switching (MPLS) label, IPv4 address, or IPv6 address. Segment Routing can be applied in MPLS data plane by encoding 20-bits SIDs in MPLS label stack [RFC8660]. It also can be applied to IPv6 data plane by encoding a list of 128-bits SIDs in IPv6 Segment Routing Extension Header (SRH) [RFC8754].

This document extends the use of the SRH [RFC8754] to unified identifiers encoded as MPLS label or IPv4 address to support more detailed network programming and interworking between SR-MPLS and SRv6 domains.

1.1. Conventions used in this document

1.1.1. Acronyms

SR: Segment Routing

SRH: Segment Routing Extension Header

MPLS: Multiprotocol Label Switching

SR-MPLS: Segment Routing using MPLS data plane

SID: Segment Identifier

IGP: Interior Gateway Protocol

DA: Destination Address

ILM: Incoming Label Map

FEC: Forwarding Equivalence Class

FTN: FEC-to-NHLFE map

OAM: Operation, Administration, and Maintenance

TE: Traffic Engineering

SRv6: Segment Routing in IPv6

U-SID: Unified Segment Identifier

PSP: Penultimate Segment Popping

FIB: Forwarding Information Base

UET: U-SID Encapsulation Type

UEC: U-SID Encapsulation Capability

1.1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Segment Routing Extension Header: Benefits and Challenges

Many functions related to Operation, Administration, and Maintenance (OAM) require identification of the SR tunnel ingress and the path, constructed by segments, between the ingress and the egress SR nodes. Combination of IPv6 encapsulation [RFC8200] and SRH [RFC8754], referred to as SRv6, comply with these requirements while it is challenging when applying SR in MPLS networks, also referred to as SR-MPLS.

On the other hand, the size of IPv6 SID presents a scaling challenge to use topological instructions that define strict explicit traffic-engineered (TE) path or support network programming in combination with service-based instructions. At the same time, that is where SR-MPLS approach provides better results due to the smaller SID length. It can be used to compress the SRv6 header size when a smaller namespace of available SIDs is sufficient for addressing the particular network.

SR-MPLS is broadly used in metro networks. With the gradual deployment of SRv6 in the core networks, supporting interworking between SR-MPLS and SRv6 becomes necessary for operators. It is operationally more efficient and straightforward if SRv6 can use the same size SIDs as in SR-MPLS. The SRH can be extended to define the same, as in SR-MPLS, SID length to support the unified segment identifier (U-SID). As a result, end-to-end SR tunnel may use U-SIDs across SR-MPLS and SRv6 domains.

3. Unified SIDs in IPv6 Segment Routing Extension Header

SRH format has been defined in Section 3 of [RFC8754] as presented in Figure 1

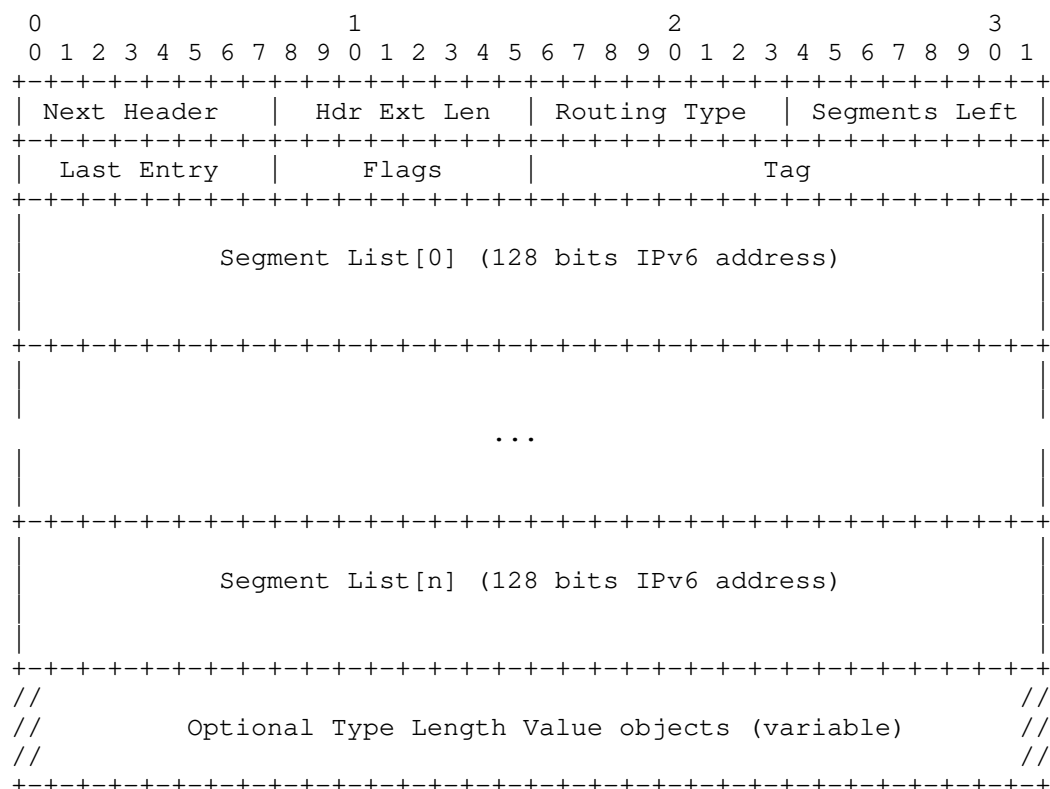


Figure 1: SRH format

This document defines a new field Size in the SRH Flags field as a two-bits field, termed as UET (U-SID Encapsulation Type) flag, to indicate which type of SIDs are encoded in SRH. The UET flag has the following values:

0b00 - indicate a 128-bits SID, an IPv6 address, termed as UET-128 U-SID.

0b01 - indicate a 32-bits SID, termed as UET-32 U-SID. In some environments, the context could be of IPv4 address, while in some other cases, it could represent an index of list or range of IPv4/IPv6 addresses. Another interpretation of 32-bits SID could be as a complementary element of an IPv4/IPv6 prefix. Setting the interpretation might be made through the control plane-based signaling and is outside the scope of this document. If this SID represents a complementary part of an IPv4/IPv6 prefix, the original IP address can be re-constructed by using, for example,

mapping, stitching, shifting, or translating operation. Specification of such a mechanism is outside the scope of this document.

0b10 - indicate a 32-bits SID, termed as UET-MPLS U-SID, which includes an MPLS label in the leftmost 20-bits as displayed in Figure 2. Information in the Context field could be interpreted as a flavor of a particular network programming behavior. Specification of the network programming using this type of U-SID is outside the scope of this document. [Ed.note. Replace with reference to the U-SID network programming document.]

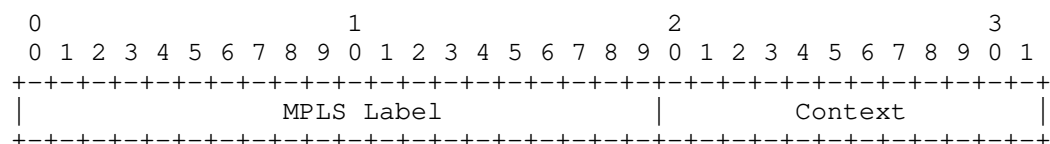


Figure 2: Format of Unified SID with MPLS Label

0b11 - indicate a 16-bits SID, termed as UET-16 U-SID. It is similar to 32bits SID and suitable for scenes with higher compression efficiency

This document also introduces a compatible operation on Segment Left field, also termed as SRH.SL. The relationship between the value of SRH.UET and the interpretation of the SRH.SL is as follows:

- * if SRH.UET Flag is UET-128, SRH.SL represents the count of 128bits-SID entries in SRH;
- * if SRH.UET Flag is UET-32 or UET-MPLS, SRH.SL represents the count of 32bits-SID entries in SRH;
- * if SRH.UET Flag is UET-16, SRH.SL represents the count of 16bits-SID entries in SRH.

4. Operations with Unified Segment Identifier

When SRH is used to include 32-bits long U-SIDs, the ingress and transit nodes of an SR tunnel act as described in Section 5.1 and Section 5.2 of [RFC8754] respectively.

4.1. Procedures of 32bits MPLS Label within SRH

This section describes how the UET-MPLS type of U-SID is used to encode a compressed SRH. In this case, an ILM (Incoming Label Map) entry can be used to map a U-SID to an IPv6 address. As a result, it is not necessary to introduce a new type of index-based mapping table. For the ILM entry of Adjacency-SID, the mapping result copied to DA (Destination Address) is the remote interface IPv6 address. For the ILM entry of Node-SID, the mapping result copied into DA is a remote node loopback IPv6 address.

Operations on an MPLS label of U-SID type are the same as those defined in [RFC8663]. However, SR-MPLS over SRH has the following advantages compared with SR-MPLS over UDP:

- * SRH is flexible to extend flags or sub-TLVs for service requirements, but not in the case of SR-MPLS over UDP.
- * Labels in SRH can meet 8 bytes alignment requirements as per [RFC8200], but not in the case of SR-MPLS over UDP.
- * The source address and the complete path information of the SR policy are not discarded, but not in the case of SR-MPLS over UDP.
- * The forwarding performance of SR-MPLS over SRH is better than the UDP method because it only updates the destination address rather than frequently removing and adding outer headers.

Procedures of SR-MPLS over IP of [RFC8663] described how to construct an adjusted SR-MPLS FTN (FEC-to-NHLFE map) and ILM entry towards a prefix-SID when next-hops are IP-only routers. The action of FTN and ILM entry will steer the packet along an outer tunnel to the destination node that has originated the FEC (Forwarding Equivalence Class). UDP header is removed and put again at each segment endpoint. However, for SR-MPLS over SRH in this document, the proposed solution is not dependent on adjusted FIB (Forwarding Information Base) entry. That is because no action is needed to get from the FIB entry. A traditional ILM entry (maybe without out-label because of IP-only next-hop) is enough to get the FEC information, i.e., map a U-SID to an IPv6 address and copy to DA. Note that an implementation can get FEC and next-hop/interface forwarding information from the ILM entry, avoiding extra FIB lookup. An SRv6 policy chosen to encapsulate the U-SID list within SRH is determined at the ingress node of this SRv6 policy. The SRH is preserved along the SR to the egress. However, in the case of PSP (Penultimate Segment Popping), the behavior is different from SR-MPLS over IP/UDP method [RFC8663], so the source address (i.e., the ingress of the SRv6 policy) is not discarded.

4.1.1. Packet Forwarding Based on UET-MPLS U-SID

The packet forwarding based on UET-MPLS U-SID is similar to the processing described in [RFC8663]. But it differs from that in FIB action and segment list processing. For completeness, we repeat the description of [RFC8663] with modification as follows.

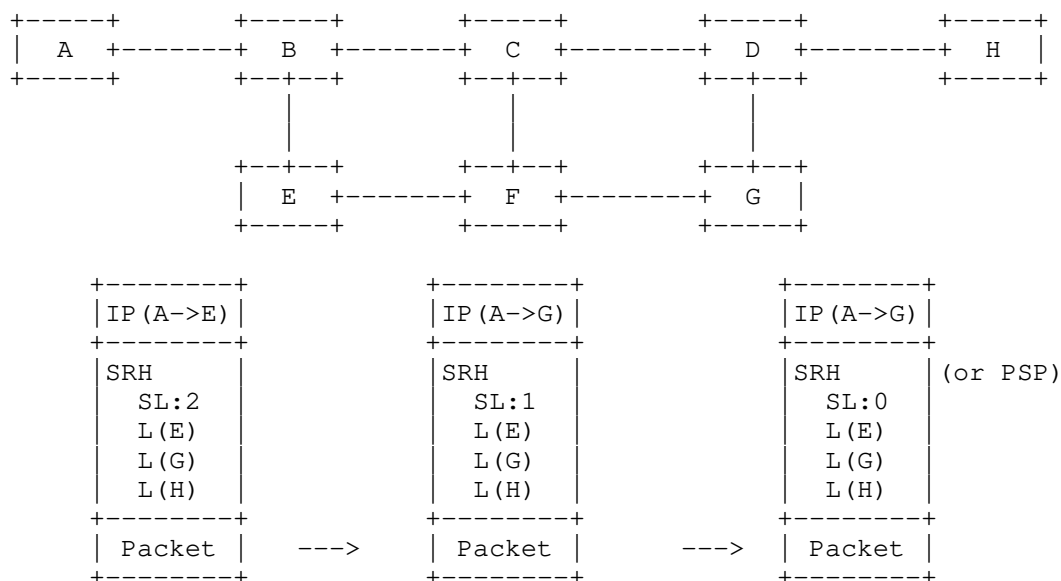


Figure 3: Packet Forwarding Example with UET-MPLS U-SID

In the example shown in Figure 3, assume that routers A, E, G, and H are U-SID capable (i.e., both SR-MPLS and SRv6 capable) while the remaining routers (B, C, D, and F) are only capable of forwarding IP packets. Routers A, E, G, and H advertise their Segment Routing related information via IS-IS or OSPF.

Now assume that router A (the Domain ingress) wants to send a packet to router H (the Domain egress) via an SRv6 policy with the explicit path {E->G->H}. Router A will impose an MPLS label stack within SRH on the packet corresponding to the explicit path. Router A searches ILM entry by the top label (that indicated router E), get the FEC information and next-hop/interface forwarding information, a loopback IPv6 address of E, and then copies to DA and sends the packet. SRH.UET is set to UET-MPLS and the value of SRH.SL is 2.

When the IPv6 packet arrives at router E, router E picks the next segment (label) within SRH based on the SRH.SL value of 2, searches ILM entry by the next label, get the FEC information and next-hop/interface forwarding information, a loopback IPv6 address of G, and then copy to DA and sends the packet. SRH.UET is set to UET-MPLS, and the value of SRH.SL is 1.

When the IPv6 packet arrives at router G, router G gets the next segment (label) within SRH based on the SRH.SL value of 1, then looks up ILM entry by the next label, gets the FEC information and next-hop/interface forwarding information, a loopback IPv6 address of H, and then copies it to IP DA and transmits the packet. Because the value of SRH.SL is 0; the SRH can be removed if the behavior flavor codepoint of the above next segment (label) is set to PSP.

4.2. Procedures of 32bits IP Address within SRH

This section describes how the UET-32 type of U-SID is used to encode a compressed SRH.

[RFC6554] specifies the Source Routing Header (to avoid confusion with Segment Routing Header, we call it SRH3 according to type 3) for use strictly between RPL (Routing Protocol for Low-Power and Lossy Networks) routers in the same RPL routing domain. It introduces mechanisms to compact the source route entries when all entries share the same prefix with the IPv6 Destination Address of a packet carrying an SRH. For each entry in Address[1..n] within the Routing header, the shared prefix octets are not carried, but only a shorter truncated piece of the original 128bits. During packet forwarding, the shorter entry gets one by one and restored to the original IPv6 address. The Segment Left field represents the number of segments remaining, i.e., the number of explicitly listed intermediate nodes still to be visited before reaching the final destination, not the number of 128bits entries.

The described above mechanism, introduced in SRH3, could also be brought to Segment Routing Header (SRH). However, unlike in SRH3, using explicit fields within the Routing header to indicate the number of prefix octets common with the IPv6 Destination Address, this document introduces a new Flavor for Endpoint Behavior, defined in [RFC8986], termed as UET Flavor, for SRv6 SIDs. The UET Flavor of the current active SID indicates the next SID's compressed length within SRH, thus preparing the next SID of the corresponding length.

The UET Flavor information of a SID can be stored in the local SID entry of that SID.

This section defines the following two UET Flavors for Endpoint Behavior:

UET-32 Flavor: a SID with UET-32 Flavor means in SRH that the next SID is a 32bits IPv4 address or number.

UET-16 Flavor: a SID with UET-16 Flavor means in SRH that the next SID is a 16bits address or number.

For the convenience of expression, we can use UET-128 Flavor for the case when the next SID is a traditional 128bits IPv6 address. Note that UET-128 Flavor is not defined in the document.

An SRv6 SID MUST NOT have multiple UET Flavors at the same time.

4.2.1. Packet Forwarding Based on UET-32 U-SID

This section describes the packet forwarding based on UET-32 U-SID. For UET-16 U-SID, it is similar.

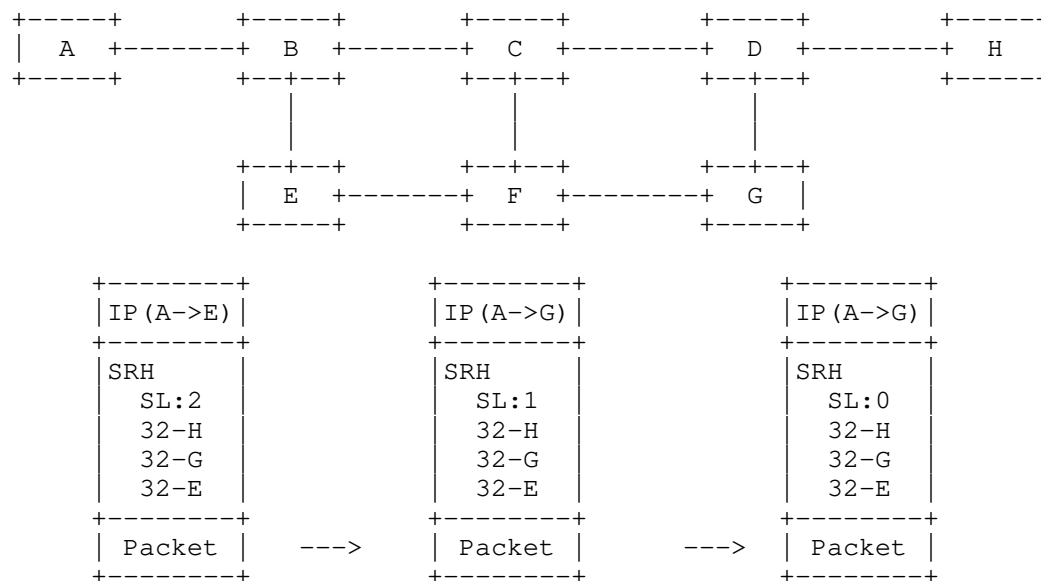


Figure 4: Packet Forwarding Example with UET-32 U-SID

In the example shown in Figure 4, assume that routers A, E, G, and H are U-SID capable while the remaining routers (B, C, D, and F) are only capable of forwarding IP packets. Routers A, E, G, and H

advertise their Segment Routing related information via IS-IS or OSPF, especially SRv6 SIDs with SID structure and UET-32 Flavor information.

Suppose that router A allocates an END SID B:32-A::, router E allocates an END SID B:32-E::, router G allocates an END SID B:32-G::, and router H allocates an END SID B:32-H::. All these SIDs have the same SID structure, i.e., share the same common prefix B (also known as the SRv6 SID Locator Block), and the sum of the Node Length, Function Length, Argument Length of each SID are the same.

Now assume that router A (the Domain ingress) wants to send a packet to router H (the Domain egress) via an SRv6 policy with the explicit path {E->G->H}. Router A will impose a UET-32 U-SID stack within SRH on the packet that corresponds to that explicit path. The U-SID stack consists of three shorter 32bits UET-32 U-SIDs, which are 32-E, 32-G, 32-H. Router A gets the first U-SID 32-E from SRH and restores it to the original IPv6 address B:32-E::, then copy it to DA and sends the packet according to IPv6 FIB lookup. SRH.UET is initially set to UET-32 and the value of SRH.SL is 2.

When the IPv6 packet arrives at router E, match the local SID entry of B:32-E::. Router E get the next U-32 32-G within SRH based on the SRH.SL value of 2, and restore it to the original IPv6 address B:32-G::, then copy it to DA and sends the packet according to IPv6 FIB lookup. SRH.UET remains unchanged, and the value of SRH.SL is 1.

When the IPv6 packet arrives at router G, match the local SID entry of B:32-G::. Router G gets the next U-32 32-H within SRH based on the SRH.SL value of 1, and restore it to the original IPv6 address B:32-H::, then copy it to DA and sends the packet according to IPv6 FIB lookup. SRH.UET remains unchanged, and the value of SRH.SL is 0. The SRH can be removed if the local SID entry of B:32-G:: has PSP Flavor.

When the IPv6 packet arrives at router H, match the local SID entry of B:32-H:: and Proceed to process the next header in the packet.

5. The Use Case of Unified Segment Identifier

In addition to being used for compression, U-SID can also be used in interworking between SR-MPLS and SRv6 domains. SR-MPLS is often used in a metro network, for example, in the backhaul metro network of CMCC. If the core network uses SRv6, for example, the core network of the same operator, U-SID can be used in the SRv6 domain to interwork with SR-MPLS in the metro network to form an end-to-end SR policy or tunnel.

5.1. Nesting Interworking Between SR-MPLS and SRv6 Using Binding U-SID

SR-MPLS uses SR SIDs as MPLS labels in the MPLS stack, and the SIDs are 32-bits long. SRv6 uses SR SIDs as IPv6 extension header in SRH, and the SIDs are 128-bits long.

The type UET-MPLS of U-SID uses the same 32-bits long SIDs in the MPLS stack and SRH. Thus, four 32-bits long U-SIDs can be placed in the space of a single 128-bits long header. The encapsulation is illustrated in Figure 5.

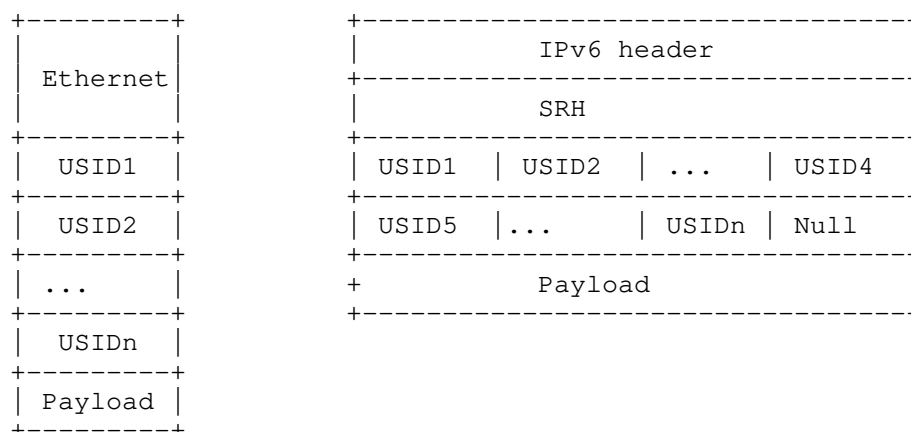


Figure 5: 32-bits long U-SIDs Encapsulation

This document RECOMMENDS using Binding SID for interworking because Binding SID allows hiding the difference between U-SID types of different domains. Additionally, a headend with only classical SRv6 SRH encapsulation capability, i.e., no capability to put multiple short U-SIDs to a single 128bits entry, will not need to upgrade.

Although Binding SID that is allocated for the specific SR policy instance will bring more states on some domain border nodes, the SR policy instance itself maybe pre-exist due to other requirements. The SR policy is created within each UET domain that can be upgraded separately.

To interwork, an MPLS Binding SID could be allocated for an SRv6 policy, used to hide the details of the UET-128 domain (classical SRv6) for a traditional MPLS Label stack. Similarly, an SRv6 Binding SID could be allocated for an SR-MPLS policy, used to hide the UET-MPLS domain's details for a conventional SRv6 SRH. An SRv6 Binding SID allocated for an SRv6 policy that enables the UET-32 compression

style will hide the details of the UET-32 domain for a traditional SRv6 SRH. There may be other combinations that are not discussed in the document.

Note that in some cases, Binding SID will cause multiple SRH to be inserted in the IPv6 header.

The SR-MPLS and SRv6 interworking is illustrated in Figure 6. An end-to-end SR path from A to F crosses the SR-MPLS and SRv6 domains. The SR-MPLS domain could be using the IPv4 or IPv6 address family. The SRv6 border nodes (E/G) receive SR-MPLS packets and forward them into the SRv6 domain using an SR-MPLS Binding SID [RFC8660].

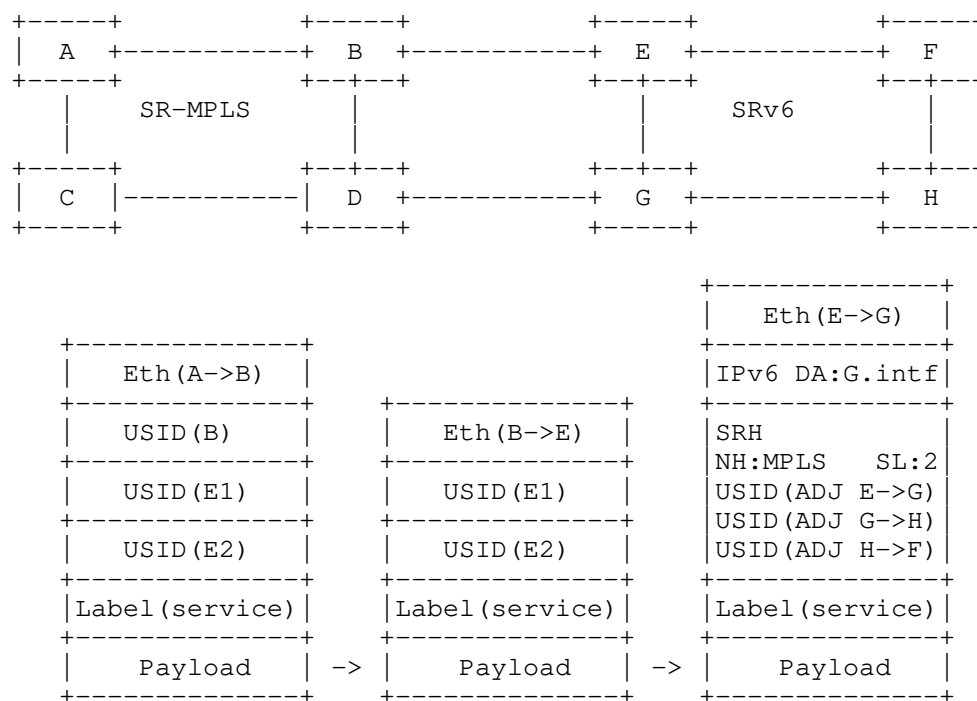


Figure 6: SR-MPLS and SRv6 interworking

The SRv6 edge node E assigns two SIDs, e.g., E1 and E2, E1 is an SR-MPLS Node-SID, E2 is an SR-MPLS Binding-SID, which represents an SRv6 policy (from E to F, via segment list E-G-H-F) with U-SID encapsulation. At the headend A, the end-to-end segment list could be B-E1-E2. Figure 3 demonstrates an example of the packet forwarding, where U-SID is an MPLS label.

The reverse interworking is illustrated in Figure 7. An end-to-end SR path from F to A crosses the SRv6 and SR-MPLS domains. The SRv6 border nodes (E/G) receive SRv6 packets and forward them into the SR-MPLS domain using an SR-MPLS Binding SID or normal Prefix/Adjacency SID.

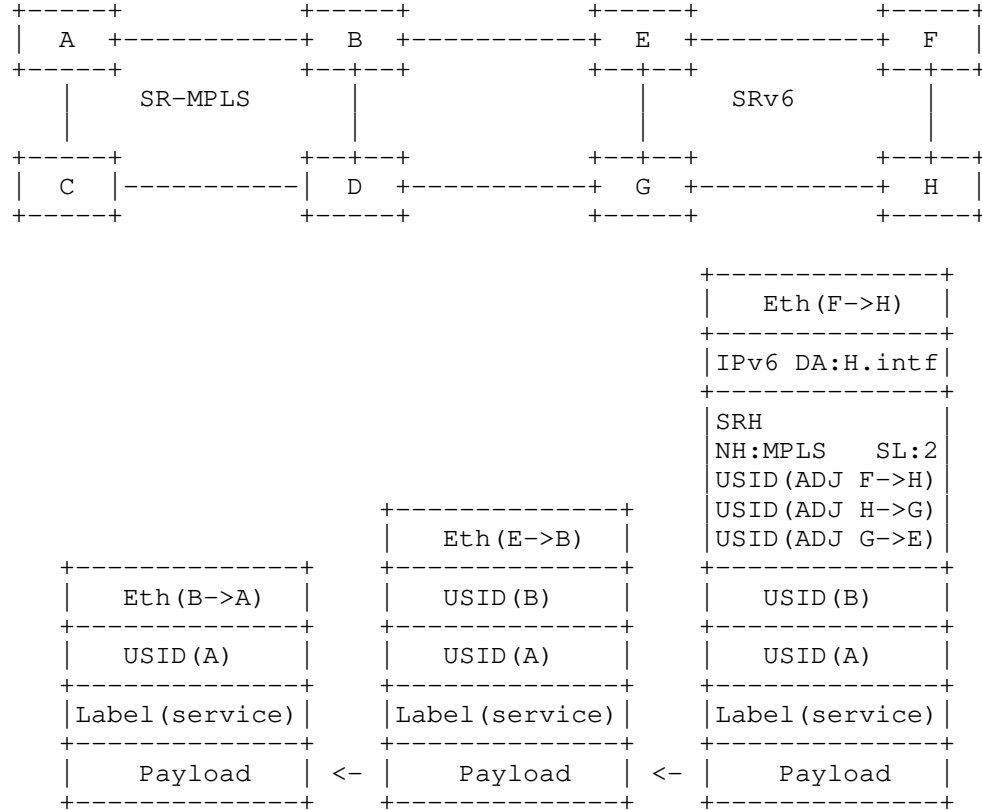


Figure 7: SR-MPLS and SRv6 reverse interworking

The SRv6 edge node F assigns an SR-MPLS Binding-SID F2, which represents an SRv6 policy (from F to E, via segment list F-H-G-E) with U-SID encapsulation. At the headend F, the end-to-end segment list could be F2-B-A.

5.2. Flat Interworking Between Different UET Domains Using Mixing U-SID

U-SRH can provide a different interworking scheme to support an end-to-end SR tunnel or policy using a mixing type of U-SIDs if more headend nodes have been upgraded to support encapsulating mixing U-SID in SRH. For example, a SID list could contain some 128bits classical SIDs, some 32bits U-SIDs (IP or Label), and some 16bits U-SIDs at the same time. For this purpose, each U-SID in SRH must meet the alignment requirement. For example, a UET-32 U-SID is stored in a 4-byte alignment, and a UET-16 U-SID is stored in a 2-byte alignment.

The interworking of different UET domains is illustrated in Figure 8. An end-to-end SR tunnel or policy from S to D with segment list <X, ABR1, Y, ABR2, Z, D>, crosses the UET-128 domain, UET-32 domain and UET-MPLS domain. Note that any order of UET domains is also possible and is similar to the case displayed in Figure 8.

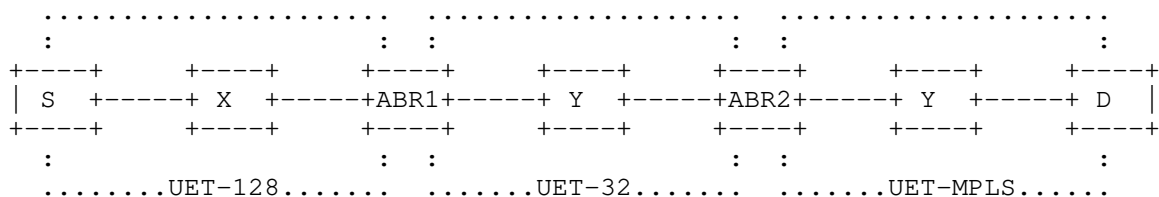


Figure 8: Interworking between different UET SID

5.2.1. UET Capability Advertisement

In an SRv6 network, each node can configure its U-SID Encapsulation Capability (UEC), and advertise it to other nodes. A controller can collect UEC information of all nodes. Typical UEC is:

UEC-128: Support classical 128bits SRv6 SID, which is the default capability of an SRv6 node.

UEC-32: Support shorter 32bits IPv4 address or number.

UEC-MPLS: Support shorter 32bits MPLS label.

UEC-16: Support shorter 16bits number.

Each node can support one or more UECs. Refer to Figure 8, node S/X/ABR1 can configure to support UEC-128 capability, node ABR1/Y/ABR2 can configure to support UEC-32 capability, and node ABR2/Y/D can configure to support UEC-MPLS capability.

A UET domain is constructed by several connected SRv6 nodes with the same UEC. For example, a UET-128 domain is constructed by the connected nodes all with UEC-128.

5.2.2. SRv6 SID Allocated per UEC

An SRv6 SID is allocated per UEC. For example, an SRv6 Node can allocate different END SIDs each for UEC-128, UEC-32, UEC-MPLS, etc.

The local SID entry of each SRv6 SID allocated per UEC will explicitly have the specific UET Flavor attribute information.

In addition to the two UET Flavors, i.e., UET-32 and UET-16 Flavors that is defined in Section 4.2, below is described a third UET Flavor for SRv6 SID:

UET-MPLS Flavor: a SID with UET-MPLS Flavor means in SRH the next SID is a 32bits MPLS label.

Each node allocates its SRv6 SID per UEC and advertises it to other nodes with additional UET-Flavor. A controller can collect these SIDs to be used for E2E SID List programming.

To save label resources, an MPLS label is not allocated per UEC. Relevant UET-Flavor information can be directly inserted in the context field of the label item in SRH. However, to meet the SRH processing restrictions defined in [RFC8754], it is possible to allocate MPLS labels for some of the topology-related SRv6 SIDs, which will consume more label resources.

Refer to the scenario presented in Figure 8, where each node may allocate the following SRv6 SID per UEC.

Node S: 128bits-END-SID-S for UEC-128.

Node X: 128bits-END-SID-X for UEC-128.

Node ABR1: 128bits-END-SID-ABR1 for UEC-128, and 128bits-END-SID-ABR1' for UEC-32.

Node Y: 128bits-END-SID-Y for UEC-32.

Node ABR2: 128bits-END-SID-ABR2 for UEC-32, and 128bits-END-SID-ABR2' for UEC-MPLS.

Node Z: 32bits-PREFIX-SID-Z. Note that MPLS Label allocation is independent with UEC.

Node D: 32bits-PREFIX-SID-D. Note that MPLS Label allocation is independent with UEC.

Note that the above SRv6 SID itself is always a 128bits IPv6 address, with no relationship with its UET Flavor attribute. The UET Flavor attribute indicates the next SID type, i.e., 128bits classical SID, 32bits IPv4 address, or 32bits MPLS Label, etc.

5.2.3. Packets Forwarding Procedures

Consider that the controller computes an E2E segment list <X, ABR1, Y, ABR2, Z, D>.

For the above E2E segment list, the controller knows which UET domain does each segment node belongs to, especially that ABR1 and ABR2 are the border nodes between different UET domains. Controller will select appropriate SID with specific UET Flavor attribute to indicate the UET domain which the next SID belongs to, i.e., whether the next SID is a classical IPv6 address or a shorter truncated value.

The SID list informed to headend could be:

FSU: First SID UET, which indicates the compression result of the first SID, in this example, is set to UET-128.

No.1 SID: 128bits-END-SID-X (with BL|TL info of itself, and UET-128 Flavor to indicate the compression result of the next SID)

No.2 SID: 128bits-END-SID-ABR1' (with BL|TL info of itself, and UET-32 Flavor to indicate the compression result of the next SID)

No.3 SID: 128bits-END-SID-Y (with BL|TL info of itself, and UET-32 Flavor to indicate the compression result of the next SID)

No.4 SID: 128bits-END-SID-ABR2' (with BL|TL info of itself, and UET-MPLS Flavor to indicate the compression result of the next SID)

No.5 SID: 32bits-PREFIX-SID-Z, (with UET-MPLS Flavor to indicate the compression result of the next SID)

No.6 SID: 32bits-PREFIX-SID-D, (with UET-128 Flavor to indicate the compression result of the next SID)

Note:

FSU indicates the compression result of the first SRv6 SID itself, while the UET Flavor attribute of the first SID just indicates the compression result of the second SRv6 SID.

BL is the Block Length of SRv6 SID. TL is the Truncated Length of SRv6 SID, i.e., the compression result.

The headend analysis of how to get the compressed SID List:

FSU is UET-128, so the first SID 128bits-END-SID-X uses 128 bits.

The No.1 SID, 128bits-END-SID-X, has UET-128 Flavor, which means the next SID, 128bits-END-SID-ABR1', also needs to use 128 bits.

The No.2 SID, 128bits-END-SID-ABR1' has UET-32 Flavor, which means the next SID, 128bits-END-SID-Y, needs to be compressed as 32 bits IPv4 address.

The No.3 SID, 128bits-END-SID-Y, has UET-32 Flavor, which means the next SID, 128bits-END-SID-ABR2', needs to be compressed as 32 bits IPv4 address.

The No.4 SID, 128bits-END-SID-ABR2', has UET-MPLS Flavor, which means the next SID, 32bits-PREFIX-SID-Z, needs to use 32 bits.

The No.5 SID, 32bits-PREFIX-SID-Z, has UET-MPLS Flavor, which means the next SID, 32bits-PREFIX-SID-D, needs to use 32 bits.

The No.6 SID, 32bits-PREFIX-SID-D, has UET-128 Flavor, which means the next SID, maybe a VPN service SRv6 SID, needs to use 128 bits.

Note that in some cases, an overlay VPN service SRv6 SID could be compressed. At that time, the last SID within the underlay segment list may select the UET-32 or UET-16 Flavor attribute.

Thus, the headend can get the following compressed SID List:

128 bits-END-SID-X with UET-128 Flavor

128 bits-END-SID-ABR1' with UET-32 Flavor

32 bits of 128 bits-END-SID-Y with UET-32 Flavor

32 bits of 128 bits-END-SID-ABR2' with UET-MPLS Flavor

32 bits-PREFIX-SID-Z (with UET-MPLS Flavor in context.field)

32 bits-PREFIX-SID-D (with UET-128 Flavor context.field)

At the However, to meet the SRH processing restrictions defined in [RFC8754], it is possible to allocate MPLS labels for some of the topology-related SRv6 SIDs, which will consume more label resources. At the headend, the encapsulated SRH could be:

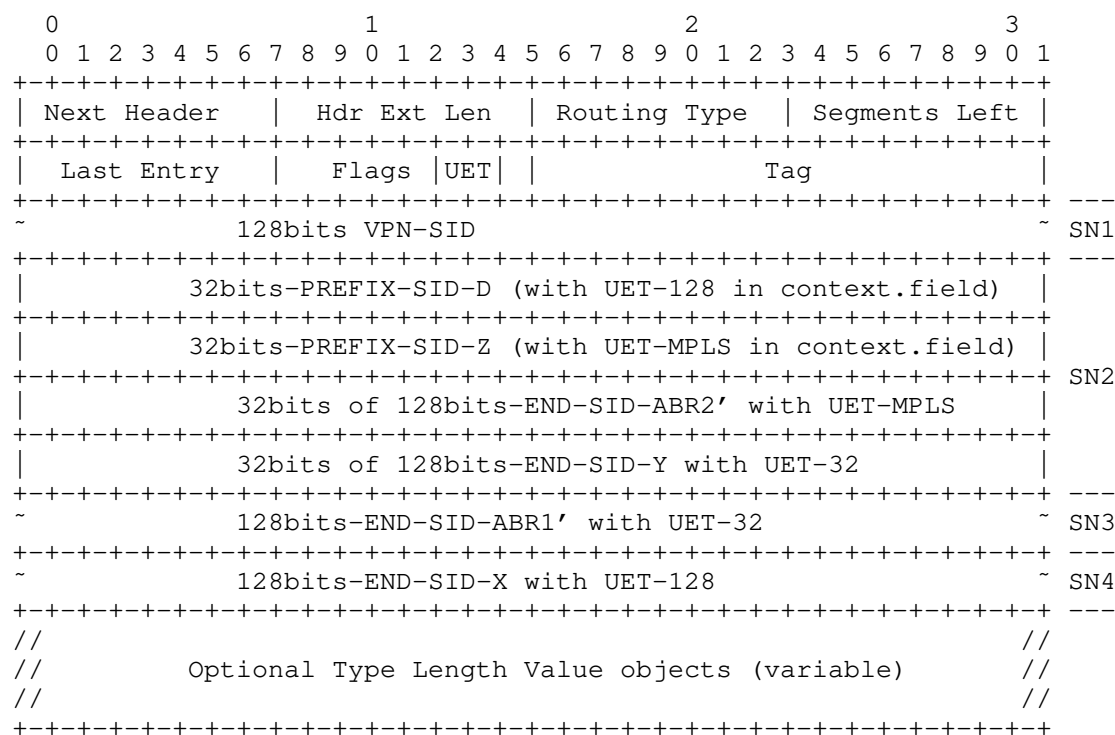


Figure 9: SRH including different UET SID

The initial SRH.SL is set to 4: the number of 128bits based SIDs in SRH, and the initial SRH.UET is set to UET-128, according to FSU, which represents the first UET domain.

During the process of packets passing through multiple UET domains, if SRH.UET change from UET-128 to UET-32 or UET-MPLS, SRH.SL will quadruple, i.e., $SRH.SL = SRH.SL * 4$, which is the number of 32bits based SIDs in SRH. When SRH.UET changed from UET-32 or UET-MPLS to UET-128, SRH.SL will revert to its original size, i.e., $SRH.SL = SRH.SL / 4$, which is the number of 128bits based SIDs in SRH.

Similarly, when SRH.UET changes from UET-128 to UET-16, $SRH.SL = SRH.SL * 8$, from UET-32 to UET-16, $SRH.SL = SRH.SL * 2$, vice versa.

Refer to Figure 8, next we will describe the process of packets passing through each UET domain.

Before transmitting packets to segment node 1 (SN1) X, the headend S decrements SRH.SL by one and gets the first 128bits SID from SRH.List[], 128bits-END-SID-X with UET-128. Then copies to DA, and then the lookups up FIB to where send the packet. Now, the SRH.SL value is 3 and SRH.UET is UET-128.

At the SN1 X, the local SID matches the DA and has the UET-128 Flavor attribute. Hence, SRH.UET has not changed. It decrements SRH.SL by one, gets the next 128bits SID from SRH.List[], 128bits-END-SID-ABR1' with UET-32, copies the value to DA, and then looks up FIB to where to send the packet. At this time, SRH.SL is 2 and SRH.UET is UET-128.

At the SN2 ABR1, the local SID matches the DA has UET-32 Flavor attribute. Hence, SRH.UET has changed from UET-128 to UET-32. The node will firstly calculate $\text{SRH.SL} * 4$, then decrement SRH.SL by 1, get the next 32bits SID from SRH.List[], 32bits of 128bits-END-SID-Y with UET-32, convert it to a complete IPv6 SID, copy to DA, and lookup FIB to send packets. At this time, SRH.SL is 7 and SRH.UET is UET-32.

At the SN3 Y, the local SID matches the DA has UET-32 Flavor attribute. Thus, SRH.UET has not changed. The node will decrement SRH.SL by 1, get the next 32bits SID from SRH.List[], 32bits of 128bits-END-SID-ABR2' with UET-MPLS, convert it to a complete IPv6 SID, copy to DA, and lookup FIB to send packets. At this time, SRH.SL is 6, SRH.UET is UET-32.

At the SN4 ABR2, the local SID matches the DA has UET-MPLS Flavor attribute. Hence, SRH.UET has changed from UET-32 to UET-MPLS. Because the size of SID has not changed, the node will decrement SRH.SL by 1, get the next 32bits SID from SRH.List[], 32bits-PREFIX-SID-Z (with UET-MPLS in context.field), map it to a complete IPv6 prefix FEC by ILM entry, copy to DA, and lookup FIB (or directly get forwarding information from ILM entry) to send packets. Note that the UET information in the context.field needs to be compared with the value in SRH.UET. Since values are equal no change and no additional processing. At this time, SRH.SL is 5, SRH.UET is 0b02.

At the SN5 Z, the address route entry matches the DA and has no UET Flavor attribute. As a result, SRH.UET has not changed. The node decrements SRH.SL by 1, will get the next 32bits SID from SRH.List[], 32bits-PREFIX-SID-D (with UET-128 in context.field), map it to a complete IPv6 prefix FEC by ILM entry, copy to DA, and lookup FIB (or directly get forwarding information from ILM entry) to send packets.

Note that the UET information in context.field needs to be compared to SRH.UET. Because it is changed from UET-MPLS to UET-128, the SRH.SL will be reverted to its original size, i.e., let $SRH.SL / 4$. At this time, SRH.SL is 1, SRH.UET is UET-128.

At the SN6 D, the address route entry matched by DA has no associated UET Flavor attribute. Hence, SRH.UET has not changed. The node decrements SRH.SL by 1, will get the next 128bits SID from SRH.List[], 128bits VPN-SID, and follow the rest process described in [RFC8986].

6. Control Plane in Support of Unified SID

The introduction of the Unified Identifier may rely on the existing SR extensions to the routing protocols. But some enhancements in the control plane are still required. This section analyzes control plane protocols and identifies necessary extensions.

Each node in the SRv6 domain needs to advertise its U-SID Encapsulation Capability, this information can be carried within SRv6-Capabilities sub-TLV defined in [I-D.ietf-lsr-isis-srv6-extensions] and SRv6 Capabilities TLV defined in [I-D.ietf-lsr-ospfv3-srv6-extensions]. It need also allocate SRv6 SID (Topology type and Service Function type) per UEC and advertise to other nodes, the advertisement of SRv6 END SID, END.X SID, LAN END.X SID defined in [I-D.ietf-lsr-isis-srv6-extensions] and [I-D.ietf-lsr-ospfv3-srv6-extensions] need to be extended to carry UET-Flavor information. This information can be collected and sent to the central controller through BGP-LS. The controller then can send the computed segment list to the headend through BGP or PCEP, and each segment will include explicit UET Flavor information. The detailed procedures are outside the scope of this document.

The SR-MPLS extensions to Interior Gateway Protocols (IGP), IS-IS [RFC8667], OSPF [RFC8665], and OSPFv3 [RFC8666], defined how 20-bits and 32-bits SIDs advertised and bound to SR objects and/or instructions. Extensions to BGP Link-state address family [RFC9085] enabled propagation of segment information of variable length via BGP. Already defined SR-MPLS extensions can be used to get MPLS U-SID mapping FIB entry, and it can coexist with SRv6 extensions to the same IGP/BGP-LS instance. For simplicity, this document suggests using the existing mature SR-MPLS control plane and FIB entry for the MPLS U-SID advertisement and mapping entry. However, it is possible to base it on SRv6 related TLVs/sub-TLVs to advertise the MPLS U-SID. It is outside the scope of this specification and will be discussed in another document.

7. SRH with U-SID Pseudo-code

Processing of SRH with U-SID is demonstrated in the following pseudo-code:

Headend sending packet:

```
S01. set initial SRH.UET, respond to the FSU, i.e.,
    the compressed result of the first SID;
S02. set initial SRH.SL, it is the count of 128bits-based SIDs;
S03. if (SRH.UET == UET-128) {
S04.   SRH.SL --;
S05.   Get SRH.List[SRH.SL], 128bits, copy to IPv6 Header DA; Or,
    headend know the first SID before SRH encapsulation,
    just copy it to DA.
S06.   FIB lookup according to DA, and forward packet;
S07. }
S08. else if (SRH.UET == UET-32) {
S09.   SRH.SL = SRH.SL * 4;
S10.   SRH.SL --;
S11.   Get SRH.List[SRH.SL], 32bits, convert to 128bits SRv6 SID, copy
    to IPv6 Header DA; Or, headend know the first SID before SRH
    encapsulation, just copy it to DA;
S12.   FIB lookup according to DA, and forward packet;
S13. }
S14. else if (SRH.UET == UET-MPLS) {
S15.   SRH.SL = SRH.SL * 4;
S16.   SRH.SL --;
S17.   Get SRH.List[SRH.SL], 32bits, lookup ILM entry and map it to 128
    IPv6 address, copy it to IPv6 Header DA; Or, headend know
    the first SID before SRH encapsulation, just copy it to DA;
S18.   FIB lookup according to DA, or directly get forwarding
    information from ILM entry and forward packet;
S19. }
S20. else if (SRH.UET == UET-16) {
S21.   SRH.SL = SRH.SL * 8;
S22.   SRH.SL --;
S23.   Get SRH.List[SRH.SL], 16bits, convert to 128bits SRv6 SID, copy
    to IPv6 Header DA; Or, headend know the first SID before SRH
    encapsulation, just copy it to DA;
S24.   FIB lookup based on DA and forward packet;
S25. }
```

Transit/Egress receive packets:

```
S01. If DA matched local SID entry, copy the UET attr of local SID entry
    to SRH.UET, check when SRH.UET changed from UET-128 to UET-32 or
        UET-MPLS, SRH.SL*4, when from UET-32 or UET-MPLS to UET-128,
        SRH.SL / 4, similar treatment of UET-16 related SRH.SL update;
    Else If DA matched normal address route entry,
        SRH.UET no update;
S02. if (SRH.SL == 0) {
S03.     process the inner payload;
S04. }
S05. else {
S06.     if (SRH.UET == UET-128) {
S07.         SRH.SL -- ;
S08.         Get SRH.List[SRH.SL], 128bits, copy it to IPv6 Header DA;
S09.         FIB lookup based on DA and forward packet;
S10.     }
S11.     else if (SRH.UET == UET-32) {
S12.         SRH.SL -- ;
S13.         Get SRH.List[SRH.SL], 32bits, convert to 128bits SRv6 SID, copy
            to IPv6 Header DA;
S14.         FIB lookup based on DA and forward packet;
S15.     }
S16.     else if (SRH.UET == UET-MPLS) {
S17.         SRH.SL --
S18.         Get SRH.List[SRH.SL], 32bits, lookup ILM entry, map it to
            128bits IPv6 address, copy it to IPv6 Header DA;
S19.         Get UET info from SRH.List[SRH.SL] Context Field, copy it to
            SRH.UET. Check if SRH.UET changed from UET-MPLS to UET-128,
            SRH.SL/4;
S20.         FIB lookup according to DA, or, directly get forwarding
            information from ILM entry, and forward packet;
S21.     }
S22.     else if (SRH.UET == UET-16) {
S23.         SRH.SL -- ;
S24.         Get SRH.List[SRH.SL], 16bits, convert to 128bits SRv6 SID, copy
            to IPv6 Header DA;
S25.         FIB lookup based on DA and forward packet;
S26.     }
S27. }
```

8. U-SID supporting SRv6 programming

U-SID can support SRv6 programming defined by [RFC8986]. The details will be described in another document.

9. Benefits

To be discussed in the next version.

10. Implementation Considerations

The Unified SID solution has been already implemented and tested by two companies:

- * Centec has conducted its PoC, and the report is available at <https://cloud.tencent.com/developer/article/1540023>.
- * Broadcom, in its lab, also conducted PoC testing of the U-SID solution.

11. IANA Considerations

IANA is requested to allocate the two-bits long field from the Segment Routing Header Flags registry referred to as Size.

12. Security Considerations

This specification inherits all security considerations of [RFC8402] and [RFC8754].

13. Contributors

Wan Xiaolan
New H3C Technologies Co. Ltd
No.8, Yongjia Road, Haidian District
Beijing
China

Email: wxlan@h3c.com

Cheng Wei
Centec
Building B, No.5 Xing Han Street, Suzhou Industrial Park
Suzhou
China

Email: Chengw@centecnetworks.com

S.Zadok
Broadcom
Israel

Email: shay.zadok@broadcom.com

14. Acknowledgements

TBD

15. Normative References

[I-D.ietf-lsr-isis-srv6-extensions]

Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over IPv6 Dataplane", Work in Progress, Internet-Draft, draft-ietf-lsr-isis-srv6-extensions-17, 18 June 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-lsr-isis-srv6-extensions-17>>.

[I-D.ietf-lsr-ospfv3-srv6-extensions]

Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", Work in Progress, Internet-Draft, draft-ietf-lsr-ospfv3-srv6-extensions-02, 15 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-lsr-ospfv3-srv6-extensions-02>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <<https://www.rfc-editor.org/info/rfc8660>>.
- [RFC8663] Xu, X., Bryant, S., Farrel, A., Hassan, S., Henderickx, W., and Z. Li, "MPLS Segment Routing over IP", RFC 8663, DOI 10.17487/RFC8663, December 2019, <<https://www.rfc-editor.org/info/rfc8663>>.
- [RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <<https://www.rfc-editor.org/info/rfc8665>>.
- [RFC8666] Psenak, P., Ed. and S. Previdi, Ed., "OSPFv3 Extensions for Segment Routing", RFC 8666, DOI 10.17487/RFC8666, December 2019, <<https://www.rfc-editor.org/info/rfc8666>>.

- [RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <<https://www.rfc-editor.org/info/rfc8667>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.
- [RFC9085] Previdi, S., Talaulikar, K., Ed., Filsfils, C., Gredler, H., and M. Chen, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing", RFC 9085, DOI 10.17487/RFC9085, August 2021, <<https://www.rfc-editor.org/info/rfc9085>>.

Authors' Addresses

Cheng Weiqiang
China Mobile
Beijing,
China

Email: chengweiqiang@chinamobile.com

Greg Mirsky
Ericsson

Email: gregimirsky@gmail.com

Peng Shaofu
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing,
China

Email: peng.shaofu@zte.com.cn

Liu Aihua
ZTE Corporation
Zhongxing Industrial Park, Nanshan District
Shenzhen,
China

Email: liu.aihua@zte.com.cn

Gyan S. Mishra
Verizon Inc.
13101 Columbia Pike
Silver Spring, MD 20904
United States of America

Phone: 301 502-1347
Email: gyan.s.mishra@verizon.com

Network Work group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

N. Nainar, Ed.
C. Pignataro, Ed.
Z. Ali
C. Filsfils
Cisco
July 8, 2019

Segment Routing Generic TLV for MPLS Label Switched Path (LSP) Ping/
Traceroute
draft-nainar-mpls-spring-lsp-ping-sr-generic-sid-00

Abstract

RFC8402 introduces Segment Routing architecture that leverages source routing and tunneling paradigms and can be directly applied to the Multi Protocol Label Switching (MPLS) data plane. A node steers a packet through a controlled set of instructions called segments, by prepending the packet with Segment Routing header. SR architecture defines different types of segments with different forwarding semantics associated. SR can be applied to the MPLS directly and to IPv6 dataplane using a new routing header.

RFC8287 defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix and IGP-Adjacency Segment Identifier (SIDs) with an MPLS data plane. Various SIDs are proposed as part of SR architecture with different associated instructions that raises a need to come up with new Target FEC Stack Sub-TLV for each such SIDs.

This document defines a new Target FEC Stack Sub-TLV that is used to validate the instruction associated with any SID.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|---|
| 1. Introduction | 2 |
| 1.1. Challenges with Existing Mechanism | 3 |
| 2. Requirements notation | 3 |
| 3. Terminology | 3 |
| 4. Target FEC Stack sub-TLV for Segment Routing SID | 4 |
| 4.1. Segment Routing Generic Label | 4 |
| 4.2. FEC for Path validation | 4 |
| 5. Procedures | 5 |
| 5.1. SID to Interface Mapping | 5 |
| 5.2. Initiator behavior | 6 |
| 5.2.1. SRGL in Target FEC Stack TLV | 6 |
| 5.3. Responder behavior | 7 |
| 5.4. PHP flag behavior | 8 |
| 6. IANA Considerations | 8 |
| 7. Security Considerations | 8 |
| 8. Acknowledgement | 8 |
| 9. Contributors | 8 |
| 10. References | 8 |
| 10.1. Normative References | 8 |
| 10.2. Informative References | 9 |
| Authors' Addresses | 9 |

1. Introduction

[RFC8402] introduces and describes a Segment Routing architecture that leverages the source routing and tunneling paradigms. A node steers a packet through a controlled set of instructions called segments, by prepending the packet with Segment Routing header. A detailed definition of the Segment Routing architecture is available in [RFC8402]

As described in [RFC8402] and [I-D.ietf-spring-segment-routing-mpls], the Segment Routing architecture can be directly applied to an MPLS data plane, the Segment identifier (Segment ID) will be of 20-bits size and the Segment Routing header is the label stack.

1.1. Challenges with Existing Mechanism

[RFC8287] defines the mechanism to perform LSP Ping and Traceroute for Segment Routing with MPLS data plane. [RFC8287] defines the Target FEC Stack Sub-TLVs for IGP-Prefix Segment ID and IGP-Adjacency Segment ID.

There are various other Segment IDs proposed by different documents that are applicable for SR architecture.

[I-D.ietf-idr-bgp-prefix-sid] defines BGP Prefix Segment ID, [I-D.ietf-idr-bgppls-segment-routing-epe] defines BGP Peering Segment ID such as Peer Node SID, Peer Adj SID and Peer Set SID.

[I-D.sivabalan-pce-binding-label-sid] defines Path Binding Segment ID. As SR evolves for different usecases, we may see more types of SIDs defined in the future. This raises a need to propose new Target FEC Stack Sub-TLV for each such Segment ID that may need specific or network wide software upgrade to support such new Target FEC Stack Sub-TLVs.

So instead of proposing different Target FEC Stack Sub-TLV for each SID, this document attempt to propose a SR Generic Label Sub-TLV for Target FEC Stack TLV with the procedure to validate the associated instruction.

This document describes the new Target FEC Stack Sub-TLV that carries the SID and the assigner node information and the procedure to use LSP Ping and Traceroute using the new sub-tlv to support path validation and fault isolation for any SR Segment IDs.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] RFC 8174 [RFC8174] when and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the terminologies defined in [RFC8402], [RFC8029], readers are expected to be familiar with it.

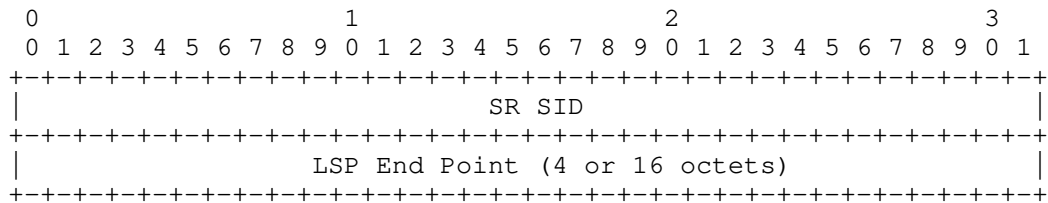
4. Target FEC Stack sub-TLV for Segment Routing SID

Following the procedure defined in [RFC8029], below defined Target FEC Stack Sub-TLV will be included for each labels in the stack. The below Sub-TLV is defined for Target FEC Stack TLV (Type 1), the Reverse-Path Target FEC Stack TLV (Type 16), and the Reply Path TLV (Type 21).

| sub-Type | Value Field |
|----------|--------------------------------------|
| TBD1 | Segment Routing Generic Label (SRGL) |

4.1. Segment Routing Generic Label

The format of the Sub-TLV is as specified below:



SR SID

Carries 20 bits of Segment ID that is used for validating the instruction.

LSP End Point

This field carries the node address of the end point that terminates the LSP.

4.2. FEC for Path validation

In SR architecture, any SID is associated with topology or service instruction. While the topology instruction steers the packet over best path or specific path, the service instruction instructs the type of service to be applied on the packet.

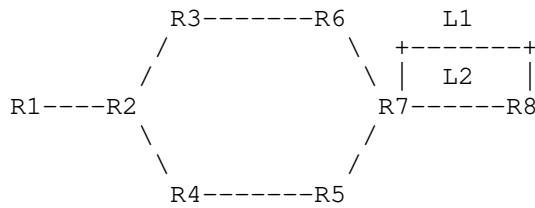


Figure 1: Segment Routing network

The Node Segment IDs for Rx for Algo 0 is 16000x. (Ex: For R1, it is 160001)
 The Node Segment IDs for Rx for Algo 128 is 16128x. (Ex: For R1, it is 161281)

9178 --> Adjacency Segment ID from R7 to R8 over link L1.
 9278 --> Adjacency Segment ID from R7 to R8 over link L2.
 9378 --> Parallel Adjacency Segment ID from R7 to R8 over Link L1 or L2.
 9187 --> Adjacency Segment ID from R8 to R7 over link L1.
 9287 --> Adjacency Segment ID from R8 to R7 over link L2.
 9387 --> Parallel Adjacency Segment ID from R8 to R7 over Link L1 or L2.

The instruction associated with any SID can be validated by verifying if the segment is terminated on the correct node and optionally received over the correct incoming interface. In Figure 1, inorder to validate the SID 9178, R1 can use {(SID=9178);(EndPoint=R8)} as FEC in Target FEC Stack Sub-TLV.

5. Procedures

This section describes the procedure to validate SR Generic Label Sub-TLV.

5.1. SID to Interface Mapping

Any End point MAY maintain a SID to Interface mapping table that maintains the below:

- o All the local Prefix/Node SID with any SR enabled interface as incoming interface.
- o All the Adj-SIDs assigned by directly connected remote nodes with the relevant interface incoming interface.

In Figure 1, R8 maintains 160008 and 161288 with Incoming interface as any SR enabled interface. Similarly, R8 maintains 9178 with Link L1 as incoming interface, 9278 with Link L2 as incoming interface and 9378 with Link L1 or L2 as incoming interface.

How this mapping is populated and maintained is a local implementation matter. It can be populated based on the IGP database or can be based on a query to Path Computation Element (PCE) controller. The mapping can be persistent or on-demand triggered by receiving LSP Ping Request.

5.2. Initiator behavior

This section defines the Target FEC Stack TLV construction mechanism by an initiator when using SR Generic Label Sub-TLV.

Ping

Initiator MUST include FEC(s) corresponding to the destination segment.

Initiator MAY include FECs corresponding to some or all of segments imposed in the label stack by the initiator to communicate the segments traversed.

Traceroute

Initiator MUST initially include FECs corresponding to all of segments imposed in the label stack.

When a received echo reply contains FEC Stack Change TLV with one or more of original segment(s) being popped, initiator MAY remove corresponding FEC(s) from Target FEC Stack TLV in the next (TTL+1) traceroute request as defined in section 4.6 of [RFC8029].

When a received echo reply does not contain FEC Stack Change TLV, initiator MUST NOT attempt to remove FEC(s) from Target FEC Stack TLV in the next (TTL+1) traceroute request.

5.2.1. SRGL in Target FEC Stack TLV

When the last segment ID in the label stack is IGP Prefix SID, Binding SID or BGP Prefix SID, set the LSP End Point field to the address of the Node that assigns the Prefix SID. The SR SID field is set to the value derived based on the index and the SRGB advertised by the LSP End Point.

When the last segment ID in the label stack is IGP Adj-SID or BGP Peering SID, set the LSP End Point field to the address of the adjacency node for which the SID is assigned to. The SR field is set to the Segment ID value.

How the above values are derived is a local implementation matter. It can be manually defined using CLI knob while triggering the LSP Ping Request or can use other mechanisms like querying the local database.

5.3. Responder behavior

Step 4a defined in Section 7.4 of [RFC8287] is updated as below:

If the Label-stack-depth is 0 and Target FEC Stack Sub-TLV at FEC-stack-depth is TBD1 (SRGL) {

- * Set the Best-return-code to 10 when LSP End Point Address does not match the local node address.
 - * Set the Best-return-code to 35, if Interface-I does not match the SID to Interface mapping for the received SR SID.
 - * set FEC-Status to 1, and return.
- }

If the Label-stack-depth is greater than 0 and Target FEC Stack Sub-TLV at FEC-stack-depth is TBD1 (SRGL), {

- * If the Label at Label-stack-depth is Imp-null {
 - + Set the Best-return-code to 10 when LSP End Point Address does not match the local node address.
 - + Set the Best-return-code to 35, if Interface-I does not match the SID to Interface mapping for the received SR SID.
 - + set FEC-Status to 1, and return.
- }
- * Else:
 - + Set the Best-return-code to 10 when the index derived from the label at Label-stack-depth is not advertised by LSP End Point.
 - + set FEC-Status to 1, and return.
- }

5.4. PHP flag behavior

To be Updated

6. IANA Considerations

To be Updated.

7. Security Considerations

To be Updated

8. Acknowledgement

TBD

9. Contributors

Danial Johari, Cisco Systems

10. References

10.1. Normative References

[I-D.ietf-idr-bgp-prefix-sid]

Previdi, S., Filsfils, C., Lindem, A., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix SID extensions for BGP", draft-ietf-idr-bgp-prefix-sid-27 (work in progress), June 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]

Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-19 (work in progress), May 2019.

[I-D.sivabalan-pce-binding-label-sid]

Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-07 (work in progress), July 2019.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <<https://www.rfc-editor.org/info/rfc8287>>.
- [RFC8402] Filts, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

10.2. Informative References

- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filts, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.

Authors' Addresses

Nagendra Kumar Nainar (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Road
Research Triangle Park, NC 27709-4987
US

Email: naikumar@cisco.com

Carlos Pignataro (editor)
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709-4987
US

Email: cpignata@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Clarence Filsfils
Cisco Systems, Inc.

Email: cfilsfil@cisco.com

Network Work group
Internet-Draft
Intended status: Standards Track
Expires: 28 May 2022

N. Nainar, Ed.
C. Pignataro, Ed.
Z. Ali
C. Filsfils
Cisco
T. Saad
Juniper
24 November 2021

Segment Routing Generic TLV for MPLS Label Switched Path (LSP) Ping/
Traceroute
draft-nainar-mpls-spring-lsp-ping-sr-generic-sid-06

Abstract

RFC8402 introduces Segment Routing architecture that leverages source routing and tunneling paradigms and can be directly applied to the Multi Protocol Label Switching (MPLS) data plane. A node steers a packet through a controlled set of instructions called segments, by prepending the packet with Segment Routing header. SR architecture defines different types of segments with different forwarding semantics associated. SR can be applied to the MPLS directly and to IPv6 dataplane using a new routing header.

RFC8287 defines the extensions to MPLS LSP Ping and Traceroute for Segment Routing IGP-Prefix and IGP-Adjacency Segment Identifier (SIDs) with an MPLS data plane. Various SIDs are proposed as part of SR architecture with different associated instructions that raises a need to come up with new Target FEC Stack Sub-TLV for each such SIDs.

This document defines a new Target FEC Stack Sub-TLV that is used to validate the instruction associated with any SID.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 May 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 1.1. Challenges with Existing Mechanism | 3 |
| 2. Requirements notation | 4 |
| 3. Terminology | 4 |
| 4. Target FEC Stack sub-TLV for Segment Routing SID | 4 |
| 4.1. Segment Routing Generic Label | 4 |
| 4.2. FEC for Path validation | 4 |
| 5. Procedures | 5 |
| 5.1. SID to Interface Mapping | 5 |
| 5.2. Initiator behavior | 6 |
| 5.2.1. SRGL in Target FEC Stack TLV | 6 |
| 5.3. Responder behavior | 7 |
| 5.4. PHP flag behavior | 7 |
| 6. IANA Considerations | 8 |
| 6.1. New Target FEC Stack Sub-TLVs | 8 |
| 6.2. Security Considerations | 8 |
| 7. Acknowledgement | 8 |
| 8. Contributors | 8 |
| 9. References | 8 |
| 9.1. Normative References | 8 |
| 9.2. Informative References | 9 |
| Authors' Addresses | 10 |

1. Introduction

[RFC8402] introduces and describes a Segment Routing architecture that leverages the source routing and tunneling paradigms. A node steers a packet through a controlled set of instructions called segments, by prepending the packet with Segment Routing header. A detailed definition of the Segment Routing architecture is available in [RFC8402]

As described in [RFC8402] and [I-D.ietf-spring-segment-routing-mpls], the Segment Routing architecture can be directly applied to an MPLS data plane, the Segment identifier (Segment ID) will be of 20-bits size and the Segment Routing header is the label stack.

1.1. Challenges with Existing Mechanism

[RFC8287] defines the mechanism to perform LSP Ping and Traceroute for Segment Routing with MPLS data plane. [RFC8287] defines the Target FEC Stack Sub-TLVs for IGP-Prefix Segment ID and IGP-Adjacency Segment ID.

There are various other Segment IDs proposed by different documents that are applicable for SR architecture.

[I-D.ietf-idr-bgp-prefix-sid] defines BGP Prefix Segment ID,

[I-D.ietf-idr-bgppls-segment-routing-epe] defines BGP Peering Segment ID such as Peer Node SID, Peer Adj SID and Peer Set SID.

[I-D.sivabalan-pce-binding-label-sid] defines Path Binding Segment ID. As SR evolves for different usecases, we may see more types of SIDs defined in the future. This raises a need to propose new Target FEC Stack Sub-TLV for each such Segment ID that may need specific or network wide software upgrade to support such new Target FEC Stack Sub-TLVs.

So instead of proposing different Target FEC Stack Sub-TLV for each SID, this document attempt to propose a SR Generic Label Sub-TLV for Target FEC Stack TLV with the procedure to validate the associated instruction.

This document describes the new Target FEC Stack Sub-TLV that carries the SID and the procedure to use LSP Ping and Traceroute using the new sub-tlv to support path validation and fault isolation for any SR Segment IDs. This document neither deprecates any existing Target FEC Stack Sub-TLVs nor precludes defining new Sub-TLVs in the future.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] RFC 8174 [RFC8174] when and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the terminologies defined in [RFC8402], [RFC8029], readers are expected to be familiar with it.

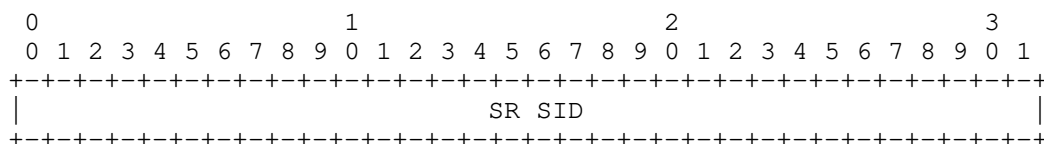
4. Target FEC Stack sub-TLV for Segment Routing SID

Following the procedure defined in [RFC8029], below defined Target FEC Stack Sub-TLV will be included for each labels in the stack. The below Sub-TLV is defined for Target FEC Stack TLV (Type 1), the Reverse-Path Target FEC Stack TLV (Type 16), and the Reply Path TLV (Type 21).

| sub-Type | Value Field |
|----------|--------------------------------------|
| TBD1 | Segment Routing Generic Label (SRGL) |

4.1. Segment Routing Generic Label

The format of the Sub-TLV is as specified below:



SR SID

Carries 20 bits of Segment ID that is used for validating the instruction.

4.2. FEC for Path validation

In SR architecture, any SID is associated with topology or service instruction. While the topology instruction steers the packet over best path or specific path, the service instruction instructs the type of service to be applied on the packet.

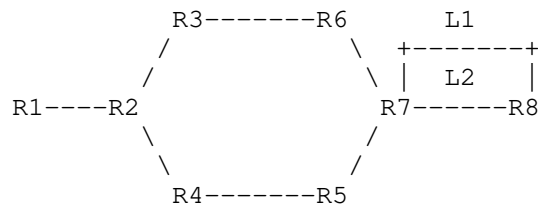


Figure 1: Segment Routing network

The Node Segment IDs for Rx for Algo 0 is 16000x. (Ex: For R1, it is 160001)
 The Node Segment IDs for Rx for Algo 128 is 16128x. (Ex: For R1, it is 161281)

9178 --> Adjacency Segment ID from R7 to R8 over link L1.
 9278 --> Adjacency Segment ID from R7 to R8 over link L2.
 9378 --> Parallel Adjacency Segment ID from R7 to R8 over Link L1 or L2.
 9187 --> Adjacency Segment ID from R8 to R7 over link L1.
 9287 --> Adjacency Segment ID from R8 to R7 over link L2.
 9387 --> Parallel Adjacency Segment ID from R8 to R7 over Link L1 or L2.

The instruction associated with any SID can be validated by verifying if the segment is terminated on the correct node and optionally received over the correct incoming interface. In Figure 1, in order to validate the SID 9178, R1 can use {(SID=9178)} as FEC in Target FEC Stack Sub-TLV.

5. Procedures

This section describes the procedure to validate SR Generic Label Sub-TLV.

5.1. SID to Interface Mapping

Any End point MAY maintain a SID to Interface mapping table that maintains the below:

- * All the local Prefix/Node SID with any SR enabled interface as incoming interface.
- * All the Adj-SIDs assigned by directly connected neighbor nodes with the relevant interface incoming interface.

In Figure 1, R8 maintains 160008 and 161288 with Incoming interface as any SR enabled interface. Similarly, R8 maintains 9178 with Link L1 as incoming interface, 9278 with Link L2 as incoming interface and 9378 with Link L1 or L2 as incoming interface.

How this mapping is populated and maintained is a local implementation matter. It can be populated based on the IGP database or can be based on a query to Path Computation Element (PCE) controller. The mapping can be persistent or on-demand triggered by receiving LSP Ping Request.

5.2. Initiator behavior

This section defines the Target FEC Stack TLV construction mechanism by an initiator when using SR Generic Label Sub-TLV.

Ping

- Initiator MUST include FEC(s) corresponding to the destination segment.
- Initiator MAY include FECs corresponding to some or all of segments imposed in the label stack by the initiator to communicate the segments traversed.

Traceroute

- Initiator MUST initially include FECs corresponding to all of segments imposed in the label stack.
- When a received echo reply contains FEC Stack Change TLV with one or more of original segment(s) being popped, initiator MAY remove corresponding FEC(s) from Target FEC Stack TLV in the next (TTL+1) traceroute request as defined in section 4.6 of [RFC8029].
- When a received echo reply does not contain FEC Stack Change TLV, initiator MUST NOT attempt to remove FEC(s) from Target FEC Stack TLV in the next (TTL+1) traceroute request.

5.2.1. SRGL in Target FEC Stack TLV

When the last segment ID in the label stack is IGP Prefix SID, Adj-SID, Binding SID, BGP Prefix SID or BGP Peering SID, set the SR SID field to the Segment ID value advertised by the LSP End Point. When the SID is advertised as index, the Segment ID value MUST be derived based on the index and the SRGB advertised by the LSP End Point.

How the above values are derived is a local implementation matter. It can be manually defined using CLI knob while triggering the LSP Ping Request or can use other mechanisms like querying the local database.

5.3. Responder behavior

Step 4a defined in Section 7.4 of [RFC8287] is updated as below:

If the Label-stack-depth is 0 and Target FEC Stack Sub-TLV at FEC-stack-depth is TBD1 (SRGL) {

- Set the Best-return-code to 10 when the responding node is not the LSP End Point for SR SID.
 - Set the Best-return-code to 35, if Interface-I does not match the SID to Interface mapping for the received SR SID.
 - set FEC-Status to 1, and return.
- }

If the Label-stack-depth is greater than 0 and Target FEC Stack Sub-TLV at FEC-stack-depth is TBD1 (SRGL), {

- If the Label at Label-stack-depth is Imp-null {
 - o Set the Best-return-code to 10 when the responding node is not the LSP End Point for the SR SID.
 - o Set the Best-return-code to 35, if Interface-I does not match the SID to Interface mapping for the received SR SID.
 - o set FEC-Status to 1, and return.
 - }
 - Else:
 - o Set the Best-return-code to 10 when the index derived from the label at Label-stack-depth is not advertised by LSP End Point.
 - o set FEC-Status to 1, and return.
- }

5.4. PHP flag behavior

Section 7.2 of [RFC8287] explains the behavior for FEC stack change for Adjacency Segment ID. The same procedure is applicable for BGP Peering SID as well.

6. IANA Considerations

6.1. New Target FEC Stack Sub-TLVs

IANA is requested to assign three new Sub-TLVs from "Sub-TLVs for TLV Types 1, 16 and 21" sub-registry from the "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" [IANA-MPLS-LSP-PING] registry.

| Sub-Type | Sub-TLV Name | Reference |
|----------|-------------------------------|------------------------------|
| TBD1 | Segment Routing Generic Label | Section 4.1 of this document |

6.2. Security Considerations

This document defines additional MPLS LSP Ping Sub-TLVs and follows the mechanisms defined in [RFC8029]. All the security considerations defined in [RFC8029] will be applicable for this document, and in addition, they do not impose any additional security challenges to be considered.

7. Acknowledgement

TBD

8. Contributors

Danial Johari, Cisco Systems

9. References

9.1. Normative References

[I-D.ietf-idr-bgp-prefix-sid]
Previdi, S., Filsfils, C., Lindem, A., Sreekantiah, A.,
and H. Gredler, "Segment Routing Prefix Segment Identifier
Extensions for BGP", Work in Progress, Internet-Draft,
draft-ietf-idr-bgp-prefix-sid-27, 26 June 2018,
<[https://www.ietf.org/archive/id/draft-ietf-idr-bgp-
prefix-sid-27.txt](https://www.ietf.org/archive/id/draft-ietf-idr-bgp-prefix-sid-27.txt)>.

- [I-D.ietf-idr-bgppls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering", Work in Progress, Internet-Draft, draft-ietf-idr-bgppls-segment-routing-epe-19, 16 May 2019, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgppls-segment-routing-epe-19.txt>>.
- [I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", Work in Progress, Internet-Draft, draft-sivabalan-pce-binding-label-sid-07, 8 July 2019, <<https://www.ietf.org/archive/id/draft-sivabalan-pce-binding-label-sid-07.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <<https://www.rfc-editor.org/info/rfc8287>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

9.2. Informative References

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with the
MPLS Data Plane", Work in Progress, Internet-Draft, draft-
ietf-spring-segment-routing-mpls-22, 1 May 2019,
<<https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-mpls-22.txt>>.

[IANA-MPLS-LSP-PING]
IANA, "Multi-Protocol Label Switching (MPLS) Label
Switched Paths (LSPs) Ping Parameters",
<<http://www.iana.org/assignments/mpls-lsp-ping-parameters/mpls-lsp-ping-parameters.xhtml>>.

Authors' Addresses

Nagendra Kumar Nainar (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Road
Research Triangle Park, NC 27709-4987
United States of America

Email: naikumar@cisco.com

Carlos Pignataro (editor)
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709-4987
United States of America

Email: cpignata@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Clarence Filsfils
Cisco Systems, Inc.

Email: cfilsfil@cisco.com

Tarek Saad
Juniper Networks

Email: tsaad@juniper.net

TEAS
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

Shaofu. Peng
Ran. Chen
Gregory. Mirsky
ZTE Corporation
Fengwei. Qin
China Mobile
November 4, 2019

Packet Network Slicing using Segment Routing
draft-peng-teas-network-slicing-01

Abstract

This document presents a mechanism aimed at providing a solution for network slicing in the transport network for 5G services. The proposed mechanism uses a unified administrative instance identifier to distinguish different virtual network resources for both intra-domain and inter-domain network slicing scenarios. Combined with the segment routing technology, the mechanism could be used for both best-effort and traffic engineered services for tenants.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Network Slicing Requirements | 3 |
| 2.1. Dedicated Virtual Networks | 3 |
| 2.2. End-to-End Slicing | 3 |
| 2.3. Unified NSI | 4 |
| 2.4. Traffic Engineering | 4 |
| 2.5. Summarized Requirements | 5 |
| 3. Conventions used in this document | 5 |
| 4. Overview of Existing Identifiers | 5 |
| 4.1. AG and EAG Bit | 6 |
| 4.2. Multi-Topology Identifier | 6 |
| 4.3. SR Policy Color | 6 |
| 4.4. Flex-algorithm Identifier | 7 |
| 4.5. New Slice-based Identifier Introduced | 7 |
| 5. Overview of AII-based Mechanism | 8 |
| 6. Resource Allocation per AII | 10 |
| 6.1. L3 Link Resource AII Configuration | 10 |
| 6.2. L2 Link Resource AII Configuration | 11 |
| 6.3. Node Resource AII Configuration | 11 |
| 7. Combined with SR Flex-algorithm for Stack Depth Optimization | 12 |
| 7.1. Best-effort Service AII-specific | 12 |
| 7.2. Traffic Engineering service AII-specific | 12 |
| 8. Examples | 13 |
| 8.1. intra-domain network slicing | 13 |
| 8.2. inter-domain network slicing via BGP-LS | 14 |
| 8.3. inter-domain network slicing via BGP-LU | 16 |
| 9. Implementation suggestions | 17 |
| 10. IANA Considerations | 17 |
| 11. Security Considerations | 18 |
| 12. Acknowledgements | 18 |
| 13. Normative references | 18 |
| Authors' Addresses | 20 |

1. Introduction

According to 5G context, network slicing is the collection of a set of technologies to create specialized, dedicated logical networks as a service (NaaS) in support of network service differentiation and meeting the diversified requirements from vertical industries. Through the flexible and customized design of functions, isolation

mechanisms, and operation and management (O&M) tools, network slicing is capable of providing dedicated virtual networks over a shared infrastructure. A Network Slice Instance (NSI) is the realization of network slicing concept. It is an E2E logical network, which comprises of a group of network functions, resources, and connection relationships. An NSI typically covers multiple technical domains, which include a terminal, access network (AN), transport network (TN) and a core network (CN), as well as a DC domain that hosts third-party applications from vertical industries. Different NSIs may have different network functions and resources. They may also share some of the network functions and resources.

For a transport network, network slicing requires the underlying network to support partitioning of the network resources to provide the client with dedicated (private) networking, computing, and storage resources drawn from a shared pool. The slices may be seen as virtual networks.

2. Network Slicing Requirements

2.1. Dedicated Virtual Networks

An end-to-end virtual network with dedicated resources is the advantage of network slicing than traditional DiffServ QoS and VPN. For example, DiffServ QoS can distinguish VoIP traffic and other type of traffic (such as high-definition video, web browsing), but can not distinguish the same type of traffic from different tenants, nor isolation of these traffic at all.

Another example is the IoT traffic of health monitoring network which connected hospital and outpatient, it always has strict privacy and safety requirements, including where the data can be stored and who can access the data, all this can not be satisfied by DiffServ QoS as it has not any function of network computing and storage.

Dedicated VN is a distinct object purchased by a customer, and it provides specific function with predictable performance, guaranteed level of isolation and safety. It is not just as QoS.

2.2. End-to-End Slicing

Only an end-to-end slice and fine-grained network can match ultra delay and safety requirements of special service. End-to-end means that it is constructed with AN-slice, TN-slice, and CN-slice part.

Although 3GPP technical specifications mainly focus on the operation and management of AN-slice and CN-slice, which include some NF (network function) components, TN-slice is also created and destroyed

according to the related NSI lifecycle. In fact, the 3GPP management system will request expected link requirements related to the network slice (e.g., topology, QoS parameters) with the help of the management system that handles the TN part related to the slice.

For TN part, the link requirements are independent of the existing domain partition of the network, i.e., any intra- or inter-domain link is the candidate resource for the slice. It is also independent of the existing underlay frame or routing technologies (IGP, BGP, Segment Routing, Flex-E, etc.), i.e., any L2 or L3 link is the candidate resource.

2.3. Unified NSI

An NSI is identified by S-NSSAI (Single Network Slice Selection Assistance Information), which is allocated per PDU session and has semantic global within the AN and CN.

For the purpose of operation and management simplicity, it is also better to have a unified identifier with semantic global to distinguish different TN-slice during the whole TN. TN-slice identifier has a mapping relation with S-NSSAI, perhaps 1:1 or 1:n.

Instead, using different slice identifier across multi-domain of TN for the specific TN-slice will introduce much and unnecessary complexity, especially for case two devices belongs to different domain try to exchange slice-based information directly, without the help of SDN controller to translate the unified TN-slice identifier to an individual domain-wide identifier.

2.4. Traffic Engineering

5G system is expected to be able to provide optimized support for a variety of different communication services, different traffic loads, and different end-user communities. For example, the communication services using network slicing may include: vehicle-to-everything (V2X) services, 5G seamless enhanced Mobile BroadBand (eMBB) service with FMC (fixed-mobile convergence), massive IoT connections. Among these service types, high data rates, high traffic densities, low-latency, high-reliability are highlighted requirements.

Traffic engineering mechanism in TN must support the above requirements, bandwidth and delay are two primary TE constraints.

2.5. Summarized Requirements

In summary, the following requirements would be satisfied:

REQ1: Provide a distinct virtual network, including dedicated topology, computation, and storage resource, not only traditional QoS;

REQ2: Unified NSI for easy operation and maintenance;

REQ3: E2E network slicing, including both intra-domain and inter-domain case;

REQ4: Customization resource for QoS purpose, bandwidth and delay are basic constraints;

REQ5: Layer 2 as well as Layer 3 link resource partition;

3. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

4. Overview of Existing Identifiers

Currently there are multiple existing mature identifiers that could be used to identify the virtual network resource in the transport network, such as:

- o Administrative Group (AG) described in [RFC3630], [RFC5329], [RFC5305] and Extended Administrative Groups (EAGs) described in [RFC7308]
- o Multi-Topology Routing (MTR) described in [RFC5120], [RFC4915], [RFC5340]
- o SR policy color described in [I-D.ietf-spring-segment-routing-policy]
- o FA-id described in [I-D.ietf-lsr-flex-algo]

However, all these identifiers are not sufficient to meet the above requirements of TN-slice. Note that all these identifiers have use case of their own, besides the network slicing use case. Next, we will discuss each of them to determine their matching of slicing requirements.

4.1. AG and EAG Bit

AG and EAG are limited to serve as a link color scheme used in TE path computation to meet the requirements of TE service for a tenant. It is difficult to use them for an NSI allocation mapping (assuming that each bit position of AG/EAG represents an NSI). Hence, they do not meet REQ1. At the same time, AG or EAG cannot be a FIB identifier for best-effort service for the same tenant.

AG and EAG are only as L3 link attribute, not appropriate for L2-bundles member, i.e., not meeting REQ5.

Note that AG and EAG have semantic global, so they meet REQ2,3.

4.2. Multi-Topology Identifier

MTR is limited to serve as an IGP logical topology scheme only used in the intra-domain scenario. Thus it is challenging to select inter-area link resources based on MT-ID when E2E inter-domain TE path needs to be created for a tenant. That is, it does not meet REQ3.

Different IGP domain within the same TN-slice may be configured with different MT-ID. Thus MT-ID does not meet REQ2.

MT-ID is only as L3 link attribute, not appropriate for L2-bundles member, so it does not meet REQ5.

4.3. SR Policy Color

The color of SR policy defines a TE purpose, which includes a set of constraints such as bandwidth, delay, TE metric, etc. Therefore color is an abstract target, and it is difficult to get a distinct virtual network according to a specific color value. In most cases, only the headend and some other border nodes need to maintain the color template, and a color-based virtual network is hard to present because of too few participants and lack of interaction scheme. That is, the color does not meet REQ1.

We can continue to define TE affinity information in color-template, but that is only appropriate for L3 link, not for L2-bundles member, so the color does not meet REQ5.

Note that the color has global semantic, so it meets REQ3.

4.4. Flex-algorithm Identifier

Indeed, FA-id is a short mapping of SR policy color, and it may inherit the matched-degree of the Policy Color. However, FA-id has its own characteristics. A specific FA-id can have more distributed participants and define explicit link resource so that an explicit FA plane can be created. Unfortunately, different best-effort and TE service of the same slice-tenant will define different constraints, resulting in the need to occupy more FA-id resources for one slice-tenant. The relationship between FA-id and slice is not clear. That is, FA-id does not meet REQ1.

On the other hand, FA-id, like MT-ID, is limited to serve as an IGP algorithm scheme used in the intra-domain scenario. It is challenging to select inter-area (especially inter-AS) link resources according to FA-id when the E2E inter-domain TE path needs to be created for the tenant. So, FA-id does not meet REQ3.

Different IGP domain within the same TN-slice may configure different FA-ids, so it does not meet REQ2.

What is more important, the path in FA plane identified by FA-id is MP2P LSP, so it is hard to define bandwidth reservation for service. So, FA-id does not meet REQ4.

The link include/exclude rules defined by FA-id is only appropriate for the L3 link, not for L2-bundles member, so FA-id does not meet REQ5.

4.5. New Slice-based Identifier Introduced

Thus, there needs to introduce a new characteristic of NSI that meets the above-listed requirements to isolate underlay resources, and it is a slice-based identifier.

Firstly, it could serve as TE criteria for TE service, this aspect is like AG/EAG; and secondly, as a FIB table identifier for best-effort service, this aspect is like MT-ID or FA-id.

This document introduces a new property of NSI called "Administrative Instance Identifier" (AII) and corresponding method of how to instantiate it in the underlay network to match the above-listed requirements.

5. Overview of AII-based Mechanism

[I-D.ietf-teas-enhanced-vpn] described a framework to create virtual networks in a packet network.

[I-D.ali-spring-network-slicing-building-blocks] described how SR policy [I-D.ietf-spring-segment-routing-policy] is competent for network slicing. This document continues to specify the detailed mechanism, according to 3GPP network slicing requirements, based on SR policy with necessary enhancement to signal association of shared resources required to create and manage an NSI and steer the packets to the path within the specific NSI.

SR policy color has semantic global in order to be conveniently exchanged between two PE routers. They configure the same color template information for the same color value. AII also with global semantic can be contained in color template to enhance SR policy to create a TE path within global TN-slice identified by AII. Besides TE service served by explicit SR policy instance, best-effort service is served by AII-specific FIB that is created by default once AII configured.

The following is how AII-based mechanism works:

At the initial stage, each link in a physical network can be colored to conform with network slicing requirements. As previously mentioned, AII can be used to color links to partition underlay resources. Also, we may continue to use AG or EAG to color links for traditional TE within a virtual network specified by an AII. A single or multiple AIIs could be configured on each intra-domain or inter-domain link regardless of IGP instance configuration. At the minimum, a link always belongs to default AII (the value is 0). The number of AIIs configured on a node's links determines the number of virtual networks the node belongs to.

The extension of the existing IGP-TE mechanisms [RFC3630] and [RFC5305] to distribute AII information in an AS as a new TE parameter of a link will be defined in another document.

An SDN controller, using BGP-LS [RFC7752] or another interface, will have a distinct view of each virtual network specified by AII. The extension of BGP-LS will also be defined in another document.

Using the CSPF algorithm, a TE path for any best-effort (BE) or traffic-engineered (TE) service can be calculated within a virtual network specified by the AII. The computation criteria could be <AII, min igp-metric> or <AII, traditional TE criteria> for the BE and TE respectively. Combined with segment routing, the TE path could be represented as:

- o a single node-SID of the destination node, for the best-effort service in the domain;
- o node-SIDs of the border node and the destination node, adjacency-SID of inter-domain link, for the inter-domain best-effort service;
- o an explicit adjacency-SID list or compressed with several loose node-SID, for P2P traffic engineered service.

Because packets of the best-effort service could be transported over an MP2P LSP without congestion control, SR best-effort FIB for each virtual network specified by AII to forward best-effort packets may be created in the IGP domain. Thus, CSPF computation with criteria $\langle \text{AII}, \text{min igp-metric} \rangle$ is distributed on each node in the IGP domain. That is similar to the behavior in [I-D.ietf-lsr-flex-algo], but the distributed CSPF computation is triggered by AII.

To distinguish forwarding behavior of different virtual networks, prefix-SID need to be allocated per AII and advertised in the IGP domain.

For inter-domain case, in addition to the destination node-SID, several node-SIDs of the domain border node and adjacency-SID of inter-domain link are also needed to construct the E2E segment list. The segment list could be computed with the help of the SDN controller, which needs to take account of AII information during the computation. The head-end of the segment list maintains the corresponding SR-TE tunnel or SR policy.

As same as the prefix-SID, adjacency-SID needs to be allocated per AII to distinguish the forwarding behavior of different virtual networks.

For P2P traffic engineering service, especially such as the ultra-reliable low-latency communication service, it SHOULD not transfer over an MP2P LSP to avoid the risk of traffic congestion. The segment list could consist of pure adjacency-SID per AII specific. The head-end of the segment list maintains the corresponding SR-TE tunnel or SR policy.

However, label stack depth of the segment list MAY be optimized at a later time based on local policies.

At this moment, we can steer traffic of overlay service to the above SR best-effort FIB, SR-TE tunnel, or SR policy instance for the specific virtual network. The overlay service could specify a color for TE purposes. For example, color 1000 means $\langle \text{AII}=10, \text{min igp-}$

metric> to say "I need best-effort forwarding within AII 10 resource", color 1001 means <AII=10, delay=10ms, AG=0x1> to say "I need traffic engineering forwarding within AII 10 resource, and only using link with AG equal to 0x1 to reach guarantee of not exceeding 10ms delay time". Service with color 1000 will be steered to an SR best-effort FIB entry, or an SR-TE tunnel/policy in case of inter-domain. Service with color 1001 will be steered to an SR-TE tunnel/policy.

Note that there is a simple variants of AII-based slicing scheme for initial slicing requirement of service, where the SDN controller in management partition the whole E2E network topology to multiple strictly isolated VNs identified by AII in local, but let the forwarding equipments be totally unaware of that. The overlay service is steered to the SR policy whose adjacency-segment list is limited within specific VN. This variants need not introduce any complex virtual network technologies to forwarding equipments, however only for limited scenes.

6. Resource Allocation per AII

6.1. L3 Link Resource AII Configuration

In IGP domain, each numbered or unnumbered L3 link could be configured with AII information and synchronized among IGP neighbors. The IGP link-state database will contain L3 links with AII information to support TE path computation taking account of AII criteria. For a numbered L3 link, it could be represented as a tuple <local node-id, remote node-id, local ip-address, remote ip-address>, for unnumbered it could be <local node-id, remote node-id, local interface-id, remote interface-id>. Each L3 link could be configured to belong to a single AII or multiple AII. Note that an L3 link always belongs to default AII(0).

For different <L3 link, AII> tuple it would allocate a different adjacency-SID, as well as advertising with different resource portion such as bandwidth occupied.

Note that AII is independent of IGP instance. An L3 link that is not part of the IGP domain, such as the special purpose for a static route, or an inter-domain link, can also be configured with AII information and allocate adjacency-SID per AII as the same as IGP links. BGP-LS could be used to collect link state data with AII information to the controller, BGP-LS has already provided a mechanism to collect link state data from many source protocols, such as IGP, Direct, Static configuration, etc., to cover network slicing requirements.

6.2. L2 Link Resource AII Configuration

[I-D.ietf-isis-l2bundles] described how to encode adjacency-SID for each L2 member link of an L3 parent link. In the network slicing scenario, it is beneficial to deploy LAG or another virtual aggregation interface between two nodes. If that, the dedicated link resources belong to different virtual networks could be added or removed on demand, they are treated as L2 member links of a single L3 virtual interface. It is the single L3 virtual interface which needs to occupy IP resource and join the IGP instance. Creating a new slice-specific link on demand or removing the old one is likely to affect little configurations.

For network slicing purpose, [I-D.ietf-isis-l2bundles] need to be extended to advertise the AII attribute for each L2 member link. For different <L2 link, AII> tuple it would allocate a different adjacency-SID, as well as advertising with different resource portion such as bandwidth occupied.

In practice, each L2 member link of an L3 parent link SUGGESTED to be configured to belong to a single AII, and different L2 member link will have different single AII configuration, with different adjacency-SID. Note that in this case, the L3 parent link belongs to default AII(0), but each L2 member link belongs to the specific non-default AII. An L2 member link maybe a Flex-E channel or UDUK tunnel created/destroyed on demand.

In the control plane, routing protocol packets following the L3 parent link will select the L2 member link with the highest priority. At the same time, in the forwarding plane, data packets that belong to the specific virtual network will pass along the L2 member link with the specific AII value.

TE path computation based on link-state database need inspect the detailed L2 members of an L3 adjacency to select the expected L2 link resource.

6.3. Node Resource AII Configuration

For topology resource, each node needs to allocate node-SID per AII when it joins the related virtual network. All nodes in the IGP domain can run the CSPF algorithm with criteria <AII, min IGP metric> to compute best-effort next-hop to any other destination nodes for a virtual network AII-specific based on the link-state database that containing AII information, so that SR best-effort FIB can be constructed for each AII. Static routes could also be added to the AII-specific FIB.

An intra-domain overlay best-effort service belongs to a virtual network could be directly matched in the SR best-effort FIB for the specific AII. At the same time, an inter-domain overlay best-effort service belongs to a virtual network could be over a segment list containing domain border node-SID and destination node-SID which could be matched in the SR best-effort FIB for the specific AII.

7. Combined with SR Flex-algorithm for Stack Depth Optimization

[I-D.ietf-lsr-flex-algo] introduced a mechanism to do label stack depth optimization for an SR policy in IGP domain part. As the color of SR policy defined a TE purpose, traditionally the headend or SDN controller will compute an expected TE path to meet that purpose. It is necessary to map a color (32 bits) to an FA-id (8 bits) when SR flex-algorithm enabled for an SR policy. Besides that, it is necessary to enable the FA-id on each node that wants to join the same FA plane manually. The FAD could copy the TE constraints (not including bandwidth case) contained in the color template. We need to consider the cost of losing the flexibility of color when executing the flex-algo optimization, and also consider the gap between P2P TE requirements and MP2P SR FA LSP capability, to reach the right balance when deciding which SR policy need optimization.

7.1. Best-effort Service AII-specific

As described above, for best-effort service we have already constructed SR best-effort FIB per AII, that is mostly like Flex-algo. Thus, it is not necessary to map to FA-id again for a color template which has defined a best-effort behavior within the dedicated AII. Of course, if someone forced to remap it, there is no downside for the operation, the overlay best-effort service (with a color which defined specific AII, best-effort requirement, and mapping FA-id) in IGP domain will try to recurse over <AII, prefix> or <FA-id, prefix> FIB entry.

7.2. Traffic Engineering service AII-specific

An SR-TE tunnel/policy that served for traffic engineering service of a virtual network specified by an AII was generated and computed according to the relevant color template, which contained specific AII and some other traditional TE constraints. If we config mapping FA-id under the color template, the SR-TE tunnel/policy instance could inherit forwarding information from corresponding SR Flex-Algo FIB entry.

8. Examples

In this section, we will further illustrate the point through some examples. All examples share the same figure below.

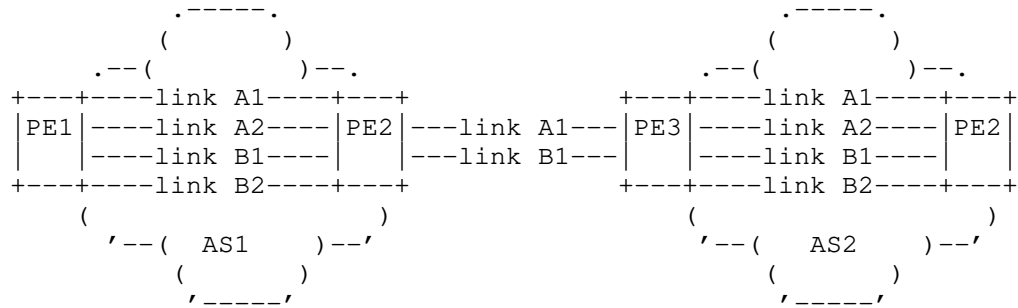


Figure 1 Network Slicing via AII

Suppose that each link belongs to separate virtual network, e.g., link Ax belongs to the virtual network colored by AII A, link Bx belongs to the virtual network colored by AII B. link x1 has an IGP metric smaller than link x2, but TE metric larger.

To simplify the use case, each AS just contained a single IGP area.

8.1. intra-domain network slicing

From the perspective of node PE1 in AS1, it will calculate best-effort forwarding entry for each AII instance (including default AII) to destinations in the same IGP area. For example:

For <AII=0, destination=ASBR1> entry, forwarding information could be ECMP during link A1 and link B1, with destination node-SID 100 for <AII=0, destination=ASBR1>.

For <AII=A, destination=ASBR1> entry, forwarding information could be link A1, with destination node-SID 200 for <AII=A, destination=ASBR1>.

For <AII=B, destination=ASBR1> entry, forwarding information could be link B1, with destination node-SID 300 for <AII=B, destination=ASBR1>.

It could also initiate an SR-TE instance (SR tunnel or SR policy) with the particular color template on PE1, PE1 is headend and ASBR1 is destination node. For example:

For SR-TE instance 1 with color template which defined criteria including {default AII, min TE metric}, forwarding information could be ECMP during two segment list {adjacency-SID 1002 for <AII=0, link A2> @PE1} and {adjacency-SID 1004 for <AII=0, link B2> @PE1}.

For SR-TE instance 2 with the color template which defined criteria including {AII=A, min TE metric}, forwarding information could be presented as the segment list {adjacency-SID 2002 for <AII=A, link A2> @PE1}.

For SR-TE instance 3 with the color template which defined criteria including {AII=B, min TE metric}, forwarding information could be presented as the segment list {adjacency-SID 3004 for <AII=B, link B2> @PE1}.

Furthermore, we can use SR Flex-algo to optimize the above SR-TE instance. For example, for SR-TE instance 1, we can define FA-ID 201 with FAD that contains the same information as the color template, in turn, FA-ID 202 for SR-TE instance 2, FA-ID 203 for SR-TE instance 3. Note that each FA-ID also needs to be enabled on ASBR1. So that the corresponding SR FA entry could be:

For <FA-ID=201, destination=ASBR1> entry, forwarding information could be ECMP during link A2 and link B2, with destination node-SID 600 for <FA-ID=201, destination=ASBR1>.

For <FA-ID=202, destination=ASBR1> entry, forwarding information could be link A2, with destination node-SID 700 for <FA-ID=202, destination=ASBR1>.

For <FA-ID=203, destination=ASBR1> entry, forwarding information could be link B2, with destination node-SID 800 for <FA-ID=203, destination=ASBR1>.

8.2. inter-domain network slicing via BGP-LS

[RFC7752] BGP-LS describes the methodology that using BGP protocol to transfer the Link-State information that maybe originated from IGP instance (for intra-domain topology information) or from local direct interface or static configuration (for inter-domain topology information). [I-D.ietf-idr-bgpls-inter-as-topology-ext] also describes a method to firstly put inter-domain interconnections to IGP instance, then always import data from IGP protocol source to

BGP-LS. In any case BGP-LS need extend to transfer the Link-State data with AII information.

An E2E inner-AS SR-TE instance with particular color template could be initiated on PE1, PE1 is head-end and PE2 is destination node. BGP-LS could be used to inform the SDN controller about the underlay network topology information including AII attribute. Thus the controller could calculate E2E TE path within the particular virtual network.

o For best-effort service, for example:

For SR-TE instance 4 with color template which defined criteria including {default AII, min IGP metric}, forwarding information could be segment list {node-SID 100 for <AII=0, destination=ASBR1> , adjacency-SID 1001 for <AII=0, link A1> @ASBR1, node-SID 400 for <AII=0, destination=PE2> }.

For SR-TE instance 5 with color template which defined criteria including {AII=A, min IGP metric}, forwarding information could be segment list {node-SID 200 for <AII=A, destination=ASBR1> , adjacency-SID 1001 for <AII=A, link A1> @ASBR1, node-SID 500 for <AII=A, destination=PE2> }.

For SR-TE instance 6 with color template which defined criteria including {AII=B, min IGP metric}, forwarding information could be segment list {node-SID 300 for <AII=B, destination=ASBR1> , adjacency-SID 1003 for <AII=B, link B1> @ASBR1, node-SID 600 for <AII=B, destination=PE2> }.

o For TE service, for example:

For SR-TE instance 7 with color template which defined criteria including {default AII, min TE metric}, forwarding information could be ECMP during two segment list {adjacency-SID 1002 for <AII=0, link A2> @PE1, adjacency-SID 1001 for <AII=0, link A1> @ASBR1, adjacency-SID 1002 for <AII=0, link A2> @ASBR2} and {adjacency-SID 1004 for <AII=0, link B2> @PE1, adjacency-SID 1003 for <AII=0, link B1> @ASBR1, adjacency-SID 1004 for <AII=0, link B2> @ASBR2}.

For SR-TE instance 8 with color template which defined criteria including {AII=A, min TE metric}, forwarding information could be segment list {adjacency-SID 2002 for <AII=A, link A2> @PE1, adjacency-SID 2001 for <AII=A, link A1> @ASBR1, adjacency-SID 2002 for <AII=A, link A2> @ASBR2}.

For SR-TE instance 9 with color template which defined criteria including {AII=B, min TE metric}, forwarding information could be

segment list {adjacency-SID 3004 for <AII=B, link B2> @PE1, adjacency-SID 3003 for <AII=B, link B1> @ASBR1, adjacency-SID 3004 for <AII=B, link B2> @ASBR2}.

For TE service, if we use SR Flex-algo to do optimization, the above forwarding information of each TE instance could inherit the corresponding SR FA entry, it would look like this:

For SR-TE instance 7, forwarding information could be ECMP during two segment list {node-SID 600 for <FA-ID=201, destination=ASBR1> , adjacency-SID 1001 for <AII=0, link A1> @ASBR1, node-SID 600 for <FA-ID=201, destination=PE2> } and {adjacency-SID 1004 for <AII=0, link B2> @PE1, adjacency-SID 1003 for <AII=0, link B1> @ASBR1, adjacency-SID 1004 for <AII=0, link B2> @ASBR2}.

For SR-TE instance 8 with color template which defined criteria including {AII=A, min TE metric}, forwarding information could be segment list {node-SID 700 for <FA-ID=202, destination=ASBR1> , adjacency-SID 2001 for <AII=A, link A1> @ASBR1, node-SID 700 for <FA-ID=202, destination=PE2> }.

For SR-TE instance 9 with color template which defined criteria including {AII=B, min TE metric}, forwarding information could be segment list {node-SID 800 for <FA-ID=203, destination=ASBR1> , adjacency-SID 3003 for <AII=B, link B1> @ASBR1, node-SID 800 for <FA-ID=203, destination=PE2> }.

8.3. inter-domain network slicing via BGP-LU

In some deployments, operators adopt BGP-LU to build inter-domain MPLS LSP, overlay service will be directly over BGP-LU LSP. If overlay service has TE requirements that defined by a color, that means that BGP-LU LSP needs to have a sense of color too, i.e., BGP-LU label could be allocated per color. At entry node of each domain, BGP-LU LSP generated for specific color will be over intra-domain SR-TE or SR Best-effort path generated for that color again. At exit node of each domain, BGP-LU LSP generated for specific color will select inter-domain forwarding resource per color. Especially, an ASBR will select slice-specific inter-AS link according to AII information of color template.

[RFC7911] defined that multiple paths UPDATE message for the same destination prefix can be advertised in BGP, each UPDATE can contain the Color Extended Community ([I-D.ietf-idr-tunnel-encaps]) with different color value.

In figure 1, PE2 can allocate and advertise six labels for its loopback plus color 1, 2, 3, 4, 5, 6 respectively. Suppose color 1

defines {default AII, min IGP metric}, color 2 defines {AII=A, min IGP metric}, color 3 defines {AII=B, min IGP metric}, and color 4 defines {default AII, min TE metric}, color 5 defines {AII=A, min TE metric}, color 6 defines {AII=B, min TE metric}. PE2 will advertise these labels to ASBR2 and ASBR2 then continues to allocate six labels each for prefix PE2 plus different color. Other nodes will have the same operation. Ultimately PE1 will maintain six BGP-LU LSP.

For example, the BGP-LU LSP for color 1 will be over SR best-effort FIB entry node-SID 100 for <AII=0, destination=ASBR1> to pass through AS1, over adjacency-SID 1001 for <AII=0, link A1>@ASBR1 to pass inter-AS, over SR best-effort FIB entry node-SID 400 for <AII=0, destination=PE2> to pass through AS2.

For example, The BGP-LU LSP for color 4 will over SR-TE instance 1 (see section 6.1), or SR best-effort FIB entry node-SID 600 for <FA-id=201, destination=ASBR1> (see section 6.1) to pass through AS1, over adjacency-SID 1001 for <AII=0, link A1>@ASBR1 to pass inter-AS, over SR-TE instance 1' or corresponding SR FA entry to pass through AS2. Note that ASBR1 need also understand the meaning of a specific color and select forwarding resource between two AS.

9. Implementation suggestions

As a node often contains control plane and forwarding plane, a suggestion is that only default AII specific FTN table, i.e, traditional FTN table, need be installed on forwarding plane, so that there are not any modification and upgrade requirement for hardware and existing MPLS forwarding mechanism. FTN entry for non-default AII instance will only be maintained on the control plane and be used for overlay service iteration according to next-hop plus color (color will give AII information and mapping FA-id information). Note that ILM entry for all AII need be installed on forwarding plane, that does not bring any confusion because of prefix-SID allocation per AII.

SR NHLFE entry and other iteration entry such as <next-hop, color> can contain AII information for expected packet scheduling.

The implementation cost is low by means of existing segment routing infrastructure.

10. IANA Considerations

TBD.

11. Security Considerations

TBD.

12. Acknowledgements

TBD.

13. Normative references

[I-D.ali-spring-network-slicing-building-blocks]

Ali, Z., Filsfils, C., Camarillo, P., and d. daniel.voyer@bell.ca, "Building blocks for Slicing in Segment Routing Network", draft-ali-spring-network-slicing-building-blocks-01 (work in progress), March 2019.

[I-D.ietf-idr-bgpls-inter-as-topology-ext]

Wang, A., Chen, H., Talaulikar, K., Zhuang, S., and S. Ma, "BGP-LS Extension for Inter-AS Topology Retrieval", draft-ietf-idr-bgpls-inter-as-topology-ext-07 (work in progress), September 2019.

[I-D.ietf-idr-tunnel-encaps]

Patel, K., Velde, G., and S. Ramachandra, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-14 (work in progress), September 2019.

[I-D.ietf-isis-l2bundles]

Ginsberg, L., Bashandy, A., Filsfils, C., Nanduri, M., and E. Aries, "Advertising L2 Bundle Member Link Attributes in IS-IS", draft-ietf-isis-l2bundles-07 (work in progress), May 2017.

[I-D.ietf-lsr-flex-algo]

Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-04 (work in progress), September 2019.

[I-D.ietf-spring-segment-routing-policy]

Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.

- [I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Networks (VPN+) Service", draft-ietf-teas-enhanced-vpn-03 (work in progress), September 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <<https://www.rfc-editor.org/info/rfc4915>>.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5329] Ishiguro, K., Manral, V., Davey, A., and A. Lindem, Ed., "Traffic Engineering Extensions to OSPF Version 3", RFC 5329, DOI 10.17487/RFC5329, September 2008, <<https://www.rfc-editor.org/info/rfc5329>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <<https://www.rfc-editor.org/info/rfc7308>>.

- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.

Authors' Addresses

Shaofu Peng
ZTE Corporation

Email: peng.shaofu@zte.com.cn

Ran Chen
ZTE Corporation

Email: chen.ran@zte.com.cn

Gregory Mirsky
ZTE Corporation

Email: gregimirsky@gmail.com

Fengwei Qin
China Mobile

Email: qinfengwei@chinamobile.com

TEAS
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2021

Shaofu. Peng
Ran. Chen
Gregory. Mirsky
ZTE Corporation
Fengwei. Qin
China Mobile
October 20, 2020

Packet Network Slicing using Segment Routing
draft-peng-teas-network-slicing-04

Abstract

This document presents a mechanism aimed at providing a solution for network slicing in the transport network for 5G services. The proposed mechanism uses a unified administrative instance identifier to distinguish different virtual network resources for both intra-domain and inter-domain network slicing scenarios. Combined with the segment routing technology, the mechanism could be used for both best-effort and traffic engineered services for tenants.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 23, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Architecture of TN Slicing | 4 |
| 2.1. Key Technologies of Transport slice | 5 |
| 3. Slicing Requirements | 6 |
| 3.1. Dedicated Virtual Networks | 6 |
| 3.2. End-to-End Slicing | 6 |
| 3.3. Unified NSI | 7 |
| 3.4. Traffic Engineering | 7 |
| 3.5. Summarized Requirements | 7 |
| 4. Conventions Used in This Document | 8 |
| 5. Overview of Existing Identifiers | 8 |
| 5.1. AG and EAG Bit | 8 |
| 5.2. Multi-Topology Identifier | 9 |
| 5.3. SR Policy Color | 9 |
| 5.4. Flex-algorithm Identifier | 9 |
| 5.5. New Slice-based Identifier Introduced | 10 |
| 6. Overview of AII-based Mechanism | 10 |
| 6.1. Physical Network Partition by AII | 11 |
| 6.2. Path within AII specific Slice | 11 |
| 6.2.1. SR-BE Path within AII specific Slice | 12 |
| 6.2.2. SR-TE Path within AII specific Slice | 13 |
| 6.3. Traffic Steering to SR policy within Slice | 13 |
| 6.4. Simple Variant of AII-based Slicing Scheme | 13 |
| 7. Resource Allocation per AII | 14 |
| 7.1. L3 Link Resource AII Configuration | 14 |
| 7.2. L2 Link Resource AII Configuration | 15 |
| 7.3. Node Resource AII Configuration | 15 |
| 7.4. Service Function Resource AII Configuration | 16 |
| 8. E2E Slicing with Centralized Mode | 16 |
| 9. E2E Slicing with Distributed Mode | 17 |
| 10. Combined with SR Flex-algorithm for Stack Depth Optimization | 17 |
| 10.1. Flex-algo Using AII Criteria | 18 |
| 10.2. Best-effort Color Template Mapping to Flex-algo | 18 |
| 10.3. Traffic Engineering Color Template Mapping to Flex-algo | 18 |
| 11. Network Slicing Examples | 18 |
| 11.1. Intra-domain Network Slicing Example | 19 |
| 11.1.1. Best-effort Service over Network Slice Example | 19 |
| 11.1.2. TE Service over Network Slice Example | 19 |
| 11.1.3. TE Service over Network Slice with Flex-algo Example | 20 |
| 11.2. Inter-domain Network Slicing via BGP-LS Example | 20 |

| | | |
|---------|---|----|
| 11.2.1. | Best-effort Service Example | 20 |
| 11.2.2. | TE Service Example | 21 |
| 11.2.3. | TE Service Using Flex-algo Example | 21 |
| 11.3. | Inter-domain Network Slicing via BGP-LU Example | 22 |
| 12. | Implementation Suggestions | 22 |
| 12.1. | SR-MPLS | 22 |
| 12.2. | SRv6 | 23 |
| 13. | IANA Considerations | 24 |
| 14. | Security Considerations | 25 |
| 15. | Acknowledgements | 26 |
| 16. | Normative references | 26 |
| | Authors' Addresses | 28 |

1. Introduction

According to 5G context, network slicing is the collection of a set of technologies to create specialized, dedicated logical networks as a service (NaaS) in support of network service differentiation and meeting the diversified requirements from vertical industries. Through the flexible and customized design of functions, isolation mechanisms, and operation and management (O&M) tools, network slicing is capable of providing dedicated virtual networks over a shared infrastructure. A Network Slice Instance (NSI) is the realization of network slicing concept. It is an E2E logical network, which comprises of a group of network functions, resources, and connection relationships. An NSI typically covers multiple technical domains, which include a terminal, access network (AN), transport network (TN) and a core network (CN), as well as a DC domain that hosts third-party applications from vertical industries. Different NSIs may have different network functions and resources. They may also share some of the network functions and resources.

For a transport network, network slicing requires the underlying network to support partitioning of the network resources to provide the client with dedicated (private) networking, computing, and storage resources drawn from a shared pool. The slices may be seen as virtual networks.

This document describes how to realize TN-slice in the underlay network, and analyze the necessity and usage of a new slice-based identifier to represent specific virtual network that is requested by the user of 5G service who have the specific resources and connection requirements. This new slice-based identifier is expected to be not only used for management system, but also for control and data plane in the underlay network.

2. Architecture of TN Slicing

Relationship with NS Design Team:

The current scope of NS design team will focus on the framework of the TN Slice. We would like to make some contributions of it, and we will sent this section to the NS Design Team for dicussion.

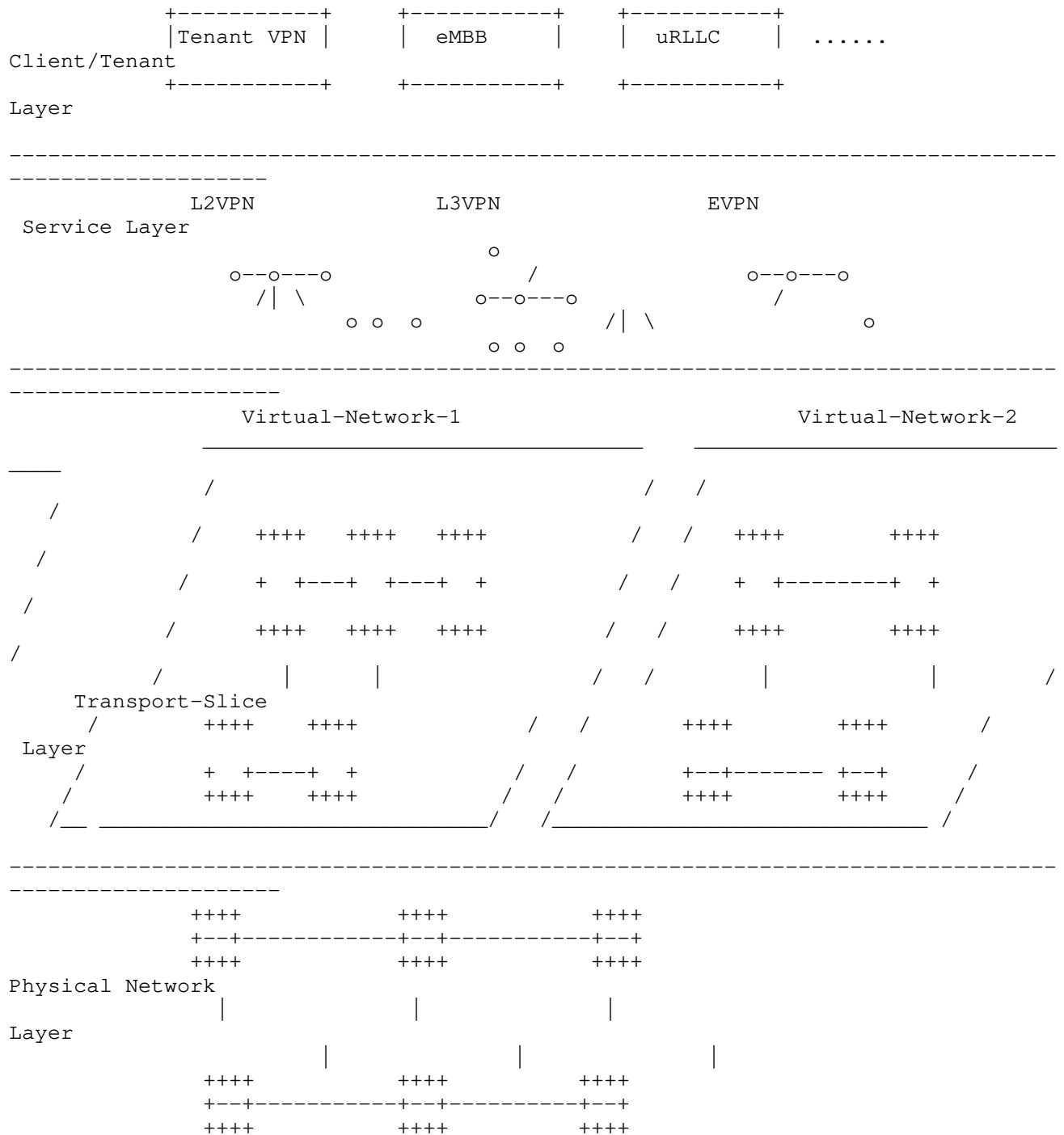


Figure 1 Architecture of TN Slicing

Based on the concept and architecture of Transport slice, the basic requirements and features of Transport slice are as following:

- o On-Demand network reconstitution: The slice network can be reconstituted in network topology and node capability to meet service needs. Each slice network has its own specific bandwidth, latency and lifecycle. Different Transport Slice networks are isolated from each other, and have independent topology and network resources.
- o Decoupling of Service Slice Layer and Physical Network Layer: The Service Slice Layer and the Physical Network Layer are decoupled, and unaware of the details of each other, which simplifies the deployment of services.
- o Similarity of Transport Slice Network and Physical Network for Service Layer: A Transport Slice Network Layer provides network resources to the upper layer (Service Layer) which is the same as the resources provided directly by a physical network from the point view of the upper layer. Services such as VPN service etc. can be deployed directly on the Transport slice network just as they are deployed on the physical network. One Transport slice network can support the deployment of more than one services or VPNs.
- o Data Plane Isolation of Transport Slice Network: The TN provides two types of traffic isolation between different TN slices: hard isolation and soft isolation. Hard isolation is implemented by providing independent circuit switched connections for the exclusive use of one slice, such as MTN (Metro Transport Network, see ITU-T G.mtn), and ODUK. Soft isolation is implemented by using a packet technology (e.g., Ethernet VLAN, MPLS tunnel, and VPN). Services of different slices are isolated from each other.
- o Transport Slice Network: There may be multiple Sub-TN-slices in a Transport Slice Network, and those Sub-Transport slices may be nested. Different sub-TN-slices can be also combined together for an end-to-end TN slice service.

2.1. Key Technologies of Transport slice

For the transport network forwarding plane slicing, there are basically two kinds of isolation technology: soft isolation technology and hard isolation technology. The soft isolation is a Layer 2 or Layer 3 technology, such as SR/IP/MPLS based tunnel technology and VPN/VLAN based virtualization technology. The hard isolation is a Layer 1 or optical-layer slicing technology based on physically rigid pipelines, such as MTN, OTN and Wavelength Division

Multiplexing (WDM) technologies. In applications, the hybrid hard and soft isolation solution is always used. The hard isolation ensures service isolation, and the soft isolation supports service bandwidth reuse.

So, The Key Technologies of Transport slice should include: Layer-one Data Plane, Layer-Two Data Plane, and Layer-Three Data Plane.

3. Slicing Requirements

3.1. Dedicated Virtual Networks

An end-to-end virtual network with dedicated resources is the advantage of network slicing than traditional DiffServ QoS and VPN. For example, DiffServ QoS can distinguish VoIP traffic and other type of traffic (such as high-definition video, web browsing), but can not distinguish the same type of traffic from different tenants, nor isolation of these traffic at all.

Another example is the IoT traffic of health monitoring network which connected hospital and outpatient, it always has strict privacy and safety requirements, including where the data can be stored and who can access the data, all this can not be satisfied by DiffServ QoS as it has not any function of network computing and storage.

Dedicated VN is a distinct object purchased by a customer, and it provides specific function with predictable performance, guaranteed level of isolation and safety. It is not just as QoS.

3.2. End-to-End Slicing

Only an end-to-end slice and fine-grained network can match ultra delay and safety requirements of special service. End-to-end means that it is constructed with AN-slice, TN-slice, and CN-slice part.

Although 3GPP technical specifications mainly focus on the operation and management of AN-slice and CN-slice, which include some NF (network function) components, TN-slice is also created and destroyed according to the related NSI lifecycle. In fact, the 3GPP management system will request expected link requirements related to the network slice (e.g., topology, QoS parameters) with the help of the management system that handles the TN part related to the slice.

For TN part, the link requirements are independent of the existing domain partition of the network, i.e., any intra- or inter-domain link is the candidate resource for the slice. It is also independent of the existing underlay frame or routing technologies (IGP, BGP,

Segment Routing, Flex-E, etc.), i.e., any L2 or L3 link is the candidate resource.

From the end-to-end slicing requirement, the inter domain resource guarantee needs to be paid more attention to.

3.3. Unified NSI

An NSI is identified by S-NSSAI (Single Network Slice Selection Assistance Information), which is allocated per PDU session and has semantic global within the AN and CN.

For the purpose of operation and management simplicity, it is also better to have a unified identifier with semantic global to distinguish different TN-slice within the whole TN. TN-slice identifier has a mapping relation with S-NSSAI, perhaps 1:1 or 1:n.

Instead, using different slice identifier across multi-domain of TN for the specific TN-slice will introduce much and unnecessary complexity, especially for case two devices belongs to different domain try to exchange slice-based information directly through the protocol mechanism in control plane, without the help of SDN controller to translate the unified TN-slice identifier to an individual domain-wide identifier.

3.4. Traffic Engineering

5G system is expected to be able to provide optimized support for a variety of different communication services, different traffic loads, and different end-user communities. For example, the communication services using network slicing may include: vehicle-to-everything (V2X) services, 5G seamless enhanced Mobile BroadBand (eMBB) service with FMC (fixed-mobile convergence), massive IoT connections. Among these service types, high data rates, high traffic densities, low-latency, high-reliability are highlighted requirements.

Traffic engineering mechanism in TN must support the above requirements, bandwidth and delay are two primary TE constraints.

3.5. Summarized Requirements

In summary, the following requirements would be satisfied for the realization of TN-slice:

REQ1: Provide a distinct virtual network, including dedicated topology, computation, and storage resource, not only traditional QoS;

REQ2: Unified NSI for easy operation and maintenance;

REQ3: E2E network slicing, including both intra-domain and inter-domain case;

REQ4: Customization resource for QoS purpose, bandwidth and delay are basic constraints;

REQ5: Layer 2 as well as Layer 3 link resource partition;

4. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

5. Overview of Existing Identifiers

Currently there are multiple existing mature identifiers that could be used to identify all or part of the virtual network's resources in the transport network, such as:

- o Administrative Group (AG) described in [RFC3630], [RFC5329], [RFC5305] and Extended Administrative Groups (EAGs) described in [RFC7308]
- o Multi-Topology Routing (MTR) described in [RFC5120], [RFC4915], [RFC5340]
- o SR policy color described in [I-D.ietf-spring-segment-routing-policy]
- o FA-id described in [I-D.ietf-lsr-flex-algo]

However, all these identifiers are not sufficient to meet the above requirements of TN-slice. Note that all these identifiers have use case of their own, besides the network slicing use case. Next, we will discuss each of them to determine their matching of slicing requirements.

5.1. AG and EAG Bit

AG and EAG are limited to serve as a link color scheme used in TE path computation to meet the requirements of TE service for a tenant. It is difficult to use them for an NSI allocation mapping (assuming that each bit position of AG/EAG represents an NSI), and also difficult to use them to represent non-link resources. Hence, they do not meet REQ1, 2. Additionally, AG or EAG cannot be as an

identifier to organize specific forwarding tables or policys for different virtual networks.

AG and EAG can be used as the attribute of both L3 interface and L2-bundles member, to meet REQ5.

Although AG and EAG have semantic global, they don't fully meet REQ3. For example, for an E2E path, inter domain links can also be selected by their AG/EAG attributes, but they can't be adhered to intra domain links through AG/EAG to let a complete view of virtual network be presented.

5.2. Multi-Topology Identifier

MTR is limited to serve as an IGP logical topology scheme only used in the intra-domain scenario. Thus it is challenging to select inter-area link resources based on MT-ID when E2E inter-domain TE path needs to be created for a tenant. And, by definition, MTR is only used to represent topology resources and cannot be used for various other types of resources. That is, it does not meet REQ1, 3.

Different IGP domain within the same TN-slice may be configured with different MT-ID. Thus MT-ID does not meet REQ2.

MT-ID is only as L3 link attribute, not appropriate for L2-bundles member, so it does not meet REQ5.

5.3. SR Policy Color

The color of SR policy defines a TE purpose, which includes a set of constraints such as bandwidth, delay, TE metric, etc. Therefore color is an abstract target, and it is difficult to get a distinct virtual network according to a specific color value. In most cases, only the headend and some other border nodes need to maintain the color template, and a color-based virtual network is hard to present because of too few participants and lack of interaction scheme. That is, the color does not meet REQ1, 2.

We can continue to define TE affinity information in color-template, to select L3 link or L2-bundles member, to meet REQ5.

Note that the color has global semantic, so it meets REQ3.

5.4. Flex-algorithm Identifier

Indeed, FA-id is a short mapping of SR policy color, and it may inherit the matched-degree of the Policy Color. However, FA-id has its own characteristics. A specific FA-id can have more distributed

participants and define explicit link resource so that an explicit FA plane can be created. Unfortunately, different service of the same tenant-slice will define different constraints, resulting in the need to occupy more FA-id resources for single tenant-slice. The relationship between FA-id and slice is not clear. That is, FA-id does not meet REQ1.

On the other hand, FA-id, like MT-ID, is limited to serve as an IGP algorithm scheme used in the intra-domain scenario. It is challenging to select inter-area (especially inter-AS) link resources according to FA-id when the E2E inter-domain TE path needs to be created for the tenant. So, FA-id does not meet REQ3.

Different IGP domain within the same TN-slice may configure different FA-ids, so it does not meet REQ2.

What is more important, the path in FA plane identified by FA-id is MP2P LSP, so it is hard to define bandwidth reservation for service. So, FA-id does not meet REQ4. Unless each link is totally dedicated to a single FA plane, i.e., link resources are not shared among multiple FA plane.

It is possible to let the link include/exclude rules defined by FA-id be appropriate for both L3 link and L2-bundles member, to meet REQ5.

5.5. New Slice-based Identifier Introduced

Thus, there needs to introduce a new characteristic of NSI that meets the above-listed requirements to isolate underlay resources, and it is a slice-based identifier.

Firstly, it could serve as TE criteria for TE service, this aspect is like AG/EAG; and secondly, as an identifier to organize specific forwarding tables or policies for different virtual networks, this aspect is like MT-ID or FA-id.

This document introduces a new property of NSI called "Administrative Instance Identifier" (AII) and corresponding method of how to instantiate it in the underlay network to match the above-listed requirements.

6. Overview of AII-based Mechanism

SR policy [I-D.ietf-spring-segment-routing-policy], just like traditional RSVP-TE LSP, can be used to realize IETF network slice in underlay network. This document introduces AII-based mechanism to enhance SR policy to support tenant-slice as well as service-slice. It will signal the association of AII and shared resources required

to create and manage an NSI, and steer the packets to the path within the specific NSI according to SR policy color.

SR policy color has semantic global in order to be conveniently exchanged between two endpoints within TN-slice. These two endpoints can configure the same color template information for the same color value. AII, also with global semantic, can be contained in color template to enhance SR policy to create a TE path within global TN-slice identified by AII. Besides TE service served by explicit SR policy instance, best-effort service can be served by AII-specific forwarding tables or policys that are created by default once AII configured.

The following is how AII-based mechanism works.

6.1. Physical Network Partition by AII

Each node and link in the physical network can be colored dynamically, with shared or dedicated resources, to conform with network slicing requirements. As previously mentioned, AII can be used to color nodes and links to partition underlay resources. Also, we may continue to use AG or EAG to color links for traditional TE within a virtual network specified by an AII. A single or multiple AIIs could be configured on each intra-domain or inter-domain link regardless of IGP instance configuration. At the minimum, a link always belongs to default AII (the value is 0).

The extension of the existing IGP-TE mechanisms [RFC3630] and [RFC5305] to distribute AII information as a new TE parameter of a link in the underlay network will be defined in another document.

Using BGP-LS [RFC7752], an SDN controller, can collect topology information of each virtual network specified by AII with related resources, and have a distinct view of each virtual network. Extension of BGP-LS will also be defined in another document.

6.2. Path within AII specific Slice

Using the CSPF algorithm, a TE path for any best-effort (BE) or traffic-engineered (TE) service can be calculated within a virtual network specified by the AII. The computation criteria could be <AII, endogenous criteria> or <AII, traditional TE critieria> for the BE and TE respectively. Combined with segment routing, the TE path could be represented as:

- o A single node-SID of the destination node, for the best-effort service intra-domain;

- o Several node-SIDs of the border node and the destination node, adjacency-SID of inter-domain link, for the inter-domain best-effort service;
- o An explicit adjacency-SID list or compressed with several loose node-SIDs, for traffic engineered service.

To distinguish forwarding behavior of different virtual networks, Prefix-SID and Adjacency-SID need to be allocated per AII with related resources, and advertised in the IGP domain.

6.2.1. SR-BE Path within AII specific Slice

Because packets of the best-effort service could be transported over an MP2P LSP without congestion control, SR-BE path for each virtual network specified by AII to forward best-effort packets may be created in the IGP domain.

This document will register some standard AII value (see section "IANA Considerations") to represent different type of slice. For example, a slice type "normal" represent any slices that have endogenous "min IGP metric" criteria to compute SPF path within the slice. Thus, for a virtual network with slice type "normal", CSPF computation with criteria <AII, min igp-metric> is distributed on each node within the virtual network.

Similarly, a slice type "uRLLC" represent any slices that have endogenous "min delay metric" (Min Unidirectional Link Delay as defined in [RFC7810]) criteria to compute SPF path within the slice. For a virtual network with slice type "uRLLC", CSPF computation with criteria <AII, min delay> is distributed on each node within the virtual network.

Similarly, a slice type "TE" represent any slices that have endogenous "TE metric" (TE default metric as defined in [RFC5305]) criteria to compute SPF path within the slice. For a virtual network with slice type "TE metric", CSPF computation with criteria <AII, min te-metric> is distributed on each node within the virtual network.

That is similar to the behavior in [I-D.ietf-lsr-flex-algo], but the distributed CSPF computation is triggered by AII, using determined criteria without negotiation among nodes.

Besides the best-effort service, SR-BE path for specific AII also provide an escape way for traffic engineering service within the same virtual network when the expected TE purpose can not be meet.

6.2.2. SR-TE Path within AII specific Slice

Other different constraints sets can also be defined to calculate TE paths for different overlay services within the same slice, regardless of the endogenous criteria of the slice. It is suggested that the set of constraints defined should contain the endogenous constraint of the slice. For some traffic engineering services that have strict requirements, especially such as the ultra-reliable low-latency communication service, it SHOULD not transfer over an MP2P LSP to avoid the risk of traffic congestion. The segment list should consist of pure adjacency-SIDs and other service function SIDs per AII, with guaranteed resources. The segment list could be computed by headend or SDN controller.

However, stack depth of the segment list MAY be optimized at a later time based on local policies.

6.3. Traffic Steering to SR policy within Slice

The traffic of overlay service can be steered to the above SR-BE path or SR-TE tunnel/policy instance for the specific virtual network. The overlay service could specify a color for TE purpose.

For example, color 1000 means <AII=10, min igp-metric> to say "I need best-effort forwarding within AII 10 resource", color 1001 means <AII=10, delay=10ms, AG=0x1> to say "I need traffic engineering forwarding within AII 10 resource, and only use links with AG 0x1 to reach guarantee of not exceeding 10ms delay upper bound".

Service with color 1000 will be steered to an SR-BE entry of the table identified by AII, or an SR-TE tunnel/policy in case of inter-domain.

Service with color 1001 will be steered to an SR-TE tunnel/policy.

6.4. Simple Variant of AII-based Slicing Scheme

There is a simple variant of AII-based slicing scheme for initial slicing requirement of service, where the SDN controller in management partition the whole E2E network topology to multiple strictly isolated VNs identified by AII in local, but let the forwarding equipments of the underlay network be totally unaware of that.

The overlay service is steered to the SR policy whose path is limited within specific VN using a pure adjacency-segment list.

This variant need not introduce any complex protocol mechanism of control plane in the underlay network, however only for limited scenes. There are no states per AII, except that QoS policy per AII need to be configured in the underlay network.

7. Resource Allocation per AII

7.1. L3 Link Resource AII Configuration

In IGP domain, each numbered or unnumbered L3 link could be configured with AII information and synchronized among IGP neighbors. The IGP link-state database will contain L3 links with AII information to support TE path computation taking account of AII criteria. For a numbered L3 link, it could be represented as a tuple <local node-id, remote node-id, local ip-address, remote ip-address>, for unnumbered it could be <local node-id, remote node-id, local interface-id, remote interface-id>. Each L3 link could be configured to belong to a single AII or multiple AII. Note that an L3 link always belongs to default AII(0).

For different <L3 link, AII> tuple it would allocate a different adjacency-SID, as well as advertising with different resource portion such as bandwidth occupied.

Note that AII is independent of IGP instance. An L3 link that is not part of the IGP domain, such as the special purpose for a static route, or an inter-domain link, can also be configured with AII information and allocate adjacency-SID per AII as the same as IGP links. BGP-LS could be used to collect link state data with AII information to the controller, BGP-LS has already provided a mechanism to collect link state data from many source protocols, such as IGP, Direct, Static configuration, etc., to cover network slicing requirements.

When an L3 link joins to multiple VNs, as traditional ways that these VNs can share the total bandwidth resource of the link with preemption mode based on packet priority, this is what we know as soft slicing. However, In some other scenarios, a hard slicing scheme can be used to establish a hardened pipe to meet the slicing business requirements, at this time each VN need dedicated bandwidth resource reserved from the same link, and at each node QoS policy per AII (e.g, traffic shaper per slice) should be used to ensure that the traffic between different slices is isolated and does not affect each other.

7.2. L2 Link Resource AII Configuration

[RFC8668] described how to encode adjacency-SID for each L2 member link of an L3 parent link. In the network slicing scenario, it is beneficial to deploy LAG or another virtual aggregation interface between two nodes. If that, the dedicated link resources belong to different virtual networks could be added or removed on demand, they are treated as L2 member links of a single L3 virtual interface. It is the single L3 virtual interface which needs to occupy IP resource and join the IGP instance. Creating a new slice-specific link on demand or removing the old one is likely to affect little configurations.

For network slicing purpose, [RFC8668] need to be extended to advertise the AII attribute for each L2 member link. In practice, for hard isolation purpose, different L2 member link of the same L3 parent link is SUGGESTED to be configured to belong to different single AII, with different adjacency-SID. Note that in this case, the L3 parent link belongs to default AII(0) (i.e, for this L3 link it is not necessary to allocate adjacency-SID per AII any more), but each L2 member link belongs to the specific non-default AII as well as the shared default AII. An L2 member link maybe a Flex-E channel or ODUK tunnel created/destroyed on demand.

In practice, for hard isolation purpose, different L2 member link of the same L3 parent link SUGGESTED to be configured to belong to different AII, with different adjacency-SID. Note that in this case, the L3 parent link belongs to default AII(0), but each L2 member link belongs to the specific non-default AII. An L2 member link maybe a Flex-E channel or ODUK tunnel created/destroyed on demand.

In the control plane, routing protocol packets following the L3 parent link will select the L2 member link with the highest priority.

In the forwarding plane, data packets that belong to the specific virtual network will pass along the L2 member link with the specific AII value.

TE path computation based on link-state database need inspect the detailed L2 members of an L3 adjacency to select the expected L2 link resource.

7.3. Node Resource AII Configuration

For topology resource, each node needs to allocate node-SID per AII when it joins the related virtual network. All nodes in the IGP domain can run the CSPF algorithm with criteria <AII, endogenous criteria> to compute SR-BE path to any other destination nodes for an

AII-specific virtual network based on the link-state database that containing AII information, so that SR-BE FIB can be constructed for each AII. Static routes could also be added to the AII-specific FIB.

An intra-domain overlay best-effort service belongs to an AII specific virtual network could be directly over the SR-BE path within that VN.

An inter-domain overlay best-effort service belongs to an AII specific virtual network could be over a path containing a single destination node-SID or a segment list containing domain border node-SID and destination node-SID within that VN.

7.4. Service Function Resource AII Configuration

[I-D.ietf-spring-sr-service-programming] introduces the notion of service segments, and describes how to implement service segments and achieve stateless service programming in SR-MPLS and SRv6 networks. The ability of encoding the service segments along with the topological segment enables service providers to forward packets along a specific network path and through VNFs or physical service appliances available in the network. Typically, a Service Function may be any purposeful execution for the packet, such as DPI, firewall, NAT, etc.

The Service Function is independent of topology, it can also be instantiated per AII, each with different priority to be executed or scheduled. For example, a docker container including specific Service Function process can be generated or destroyed on demand according to the life-cycle of a particular slice. It will have a particular CPU scheduling priority.

At a node, multiple instance of the same type of Service Function for different slice will allocate different Service SID and advertise to other nodes.

8. E2E Slicing with Centralized Mode

[RFC7752] BGP-LS describes the methodology that using BGP protocol to transfer the Link-State information that maybe originated from IGP instance (for intra-domain topology information) or from local direct interface or static configuration(for inter-domain topology information). [I-D.ietf-idr-bgppls-inter-as-topology-ext] also describes a method to firstly put inter-domain interconnections to IGP instance, then always import data from IGP protocol source to BGP-LS. In any case BGP-LS need extend to transfer the Link-State data with AII information.

An E2E inner-AS SR-TE instance with particular color template could be initiated on PE1, PE1 is head-end and PE2 is destination node. BGP-LS could be used to inform the SDN controller about the underlay network topology information including AII attribute. Thus the controller could calculate E2E TE path within the particular virtual network. Especially an AII specific Adacency-SID of inter-domain link can be included in the E2E SID list.

9. E2E Slicing with Distributed Mode

In some deployments, especially the network evolution from seamless MPLS in practice, operators adopt BGP-LU to build inter-domain MPLS LSP, and overlay service will be directly over BGP-LU LSP.

In this case, the network is divided into some domains and each domain will run its own IGP process. These IGP process are isolated to each other to be simple. That means it is inconvenient to realize network slicing depending on IGP itself with inter-area route leak or redistribution.

For an E2E BGP-LU LSP, if overlay service has TE requirements that defined by a color, the BGP-LU LSP need also have a sense of color, i.e., BGP-LU label could be allocated per color.

At entry node of each domain, BGP-LU LSP generated for specific color will be over intra-domain SR-TE or SR-BE path generated for that color again. At exit node of each domain, BGP-LU LSP generated for specific color will select inter-domain forwarding resource per color. Especially, an ASBR will select slice-specific inter-AS link according to AII information of color template.

[RFC7911] defined that multiple paths UPDATE message for the same destination prefix can be advertised in BGP, each UPDATE can contain the Color Extended Community ([I-D.ietf-idr-tunnel-encaps]) with different color value. That is a simple existing way to realize BGP-LU color function, with needless new BGP extensions.

10. Combined with SR Flex-algorithm for Stack Depth Optimization

[I-D.ietf-lsr-flex-algo] introduced a mechanism to do segment stack depth optimization for an SR policy in IGP domain part. As the color of SR policy defined a TE purpose, traditionally the headend or SDN controller will compute an expected TE path to meet that purpose.

It is necessary to map a color (32 bits) to an FA-id (8 bits) when SR flex-algorithm enabled for an SR policy. Besides that, it is necessary to enable the FA-id on each node that wants to join the

same FA plane manually. The FAD could copy the TE constraints (not including bandwidth case) contained in the color template.

We need to consider the cost of losing the flexibility of color when executing the flex-algo optimization, and also consider the gap between P2P TE requirements and MP2P SR FA LSP capability, to reach the right balance when deciding which SR policy need optimization.

10.1. Flex-algo Using AII Criteria

Because the first feature of AII is a TE criteria of link and node, it could be served as a parameter of Flex-algo Definition. [I-D.peng-lsr-flex-algo-opt-slicing] described how to extend IGP Flex-algo to compute constraint based paths over the AII specific network slice.

10.2. Best-effort Color Template Mapping to Flex-algo

As described above, for best-effort service within an AII specific virtual network, we have already constructed SR-BE FIB tables per AII, that is mostly like Flex-algo. Thus, it is not necessary to map to FA-id again for a color template which has defined a best-effort behavior within an AII specific virtual network. Of course, if someone forced to remap it, there is no downside for the operation, the overlay best-effort service (with a color which defined specific AII, best-effort requirement, and mapping FA-id) in IGP domain will try to recurse over <AII, prefix> or <FA-id, prefix> FIB entry.

10.3. Traffic Engineering Color Template Mapping to Flex-algo

An SR-TE tunnel/policy that served for traffic engineering service of an AII specific virtual network was generated and computed according to the relevant color template, which contained specific AII and some other traditional TE constraints. If we config mapping FA-id under the color template, the SR-TE tunnel/policy instance will inherit forwarding information from corresponding SR Flex-Algo FIB entry. If we config merging FA-id under the color template, the SR-TE tunnel/policy instance will have a TE path within Flex-algo plane.

11. Network Slicing Examples

In this section, we will further illustrate the point through some examples. All examples share the same figure below.

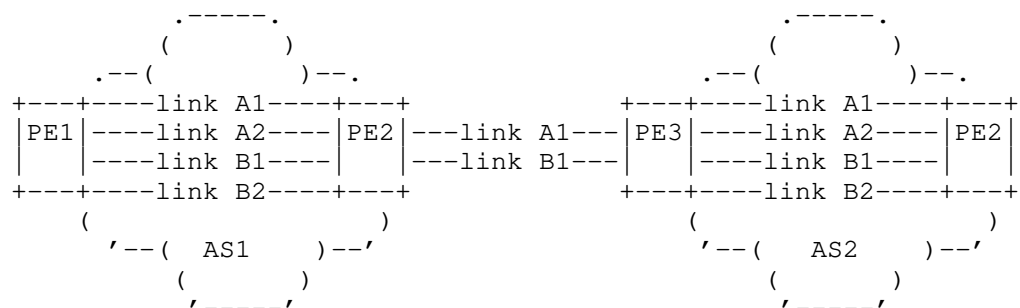


Figure 2 Network Slicing via AII

Suppose that each link belongs to separate virtual network, e.g., link Ax belongs to the virtual network colored by AII A, link Bx belongs to the virtual network colored by AII B. link x1 has an IGP metric smaller than link x2, but TE metric larger.

To simplify the use case, each AS just contained a single IGP area.

11.1. Intra-domain Network Slicing Example

11.1.1. Best-effort Service over Network Slice Example

From the perspective of node PE1 in AS1, it will calculate best-effort forwarding entry for each AII instance (including default AII) to destinations in the same IGP area. For example:

For <AII=0, destination=ASBR1> entry, forwarding information could be ECMP during link A1 and link B1, with destination node-SID 100 for <AII=0, destination=ASBR1>.

For <AII=A, destination=ASBR1> entry, forwarding information could be link A1, with destination node-SID 200 for <AII=A, destination=ASBR1>.

For <AII=B, destination=ASBR1> entry, forwarding information could be link B1, with destination node-SID 300 for <AII=B, destination=ASBR1>.

11.1.2. TE Service over Network Slice Example

It could also initiate an SR-TE instance (SR tunnel or SR policy) with the particular color template on PE1, PE1 is headend and ASBR1 is destination node. For example:

For SR-TE instance 1 with color template which defined criteria including {default AII, min TE metric}, forwarding information could be ECMP during two segment list {adjacency-SID 1002 for <AII=0, link A2> @PE1} and {adjacency-SID 1004 for <AII=0, link B2> @PE1}.

For SR-TE instance 2 with the color template which defined criteria including {AII=A, min TE metric}, forwarding information could be presented as the segment list {adjacency-SID 2002 for <AII=A, link A2> @PE1}.

For SR-TE instance 3 with the color template which defined criteria including {AII=B, min TE metric}, forwarding information could be presented as the segment list {adjacency-SID 3004 for <AII=B, link B2> @PE1}.

11.1.3. TE Service over Network Slice with Flex-algo Example

Furthermore, we can use SR Flex-algo to optimize the above SR-TE instance. For example, for SR-TE instance 1, we can define FA-ID 201 with FAD that contains the same information as the color template, in turn, FA-ID 202 for SR-TE instance 2, FA-ID 203 for SR-TE instance 3. Note that each FA-ID also needs to be enabled on ASBR1. So that the corresponding SR FA entry could be:

For <FA-ID=201, destination=ASBR1> entry, forwarding information could be ECMP during link A2 and link B2, with destination node-SID 600 for <FA-ID=201, destination=ASBR1>.

For <FA-ID=202, destination=ASBR1> entry, forwarding information could be link A2, with destination node-SID 700 for <FA-ID=202, destination=ASBR1>.

For <FA-ID=203, destination=ASBR1> entry, forwarding information could be link B2, with destination node-SID 800 for <FA-ID=203, destination=ASBR1>.

11.2. Inter-domain Network Slicing via BGP-LS Example

11.2.1. Best-effort Service Example

For SR-TE instance 4 with color template which defined criteria including {default AII, min IGP metric}, forwarding information could be segment list {node-SID 100 for <AII=0, destination=ASBR1> , adjacency-SID 1001 for <AII=0, link A1> @ASBR1, node-SID 400 for <AII=0, destination=PE2> }.

For SR-TE instance 5 with color template which defined criteria including {AII=A, min IGP metric}, forwarding information could be

segment list {node-SID 200 for <AII=A, destination=ASBR1> ,
adjacency-SID 1001 for <AII=A, link A1> @ASBR1, node-SID 500 for
<AII=A, destination=PE2> }.

For SR-TE instance 6 with color template which defined criteria
including {AII=B, min IGP metric}, forwarding information could be
segment list {node-SID 300 for <AII=B, destination=ASBR1> ,
adjacency-SID 1003 for <AII=B, link B1> @ASBR1, node-SID 600 for
<AII=B, destination=PE2> }.

11.2.2. TE Service Example

For SR-TE instance 7 with color template which defined criteria
including {default AII, min TE metric}, forwarding information could
be ECMP during two segment list {adjacency-SID 1002 for <AII=0, link
A2> @PE1, adjacency-SID 1001 for <AII=0, link A1> @ASBR1, adjacency-
SID 1002 for <AII=0, link A2> @ASBR2} and {adjacency-SID 1004 for
<AII=0, link B2> @PE1, adjacency-SID 1003 for <AII=0, link B1>
@ASBR1, adjacency-SID 1004 for <AII=0, link B2> @ASBR2}.

For SR-TE instance 8 with color template which defined criteria
including {AII=A, min TE metric}, forwarding information could be
segment list {adjacency-SID 2002 for <AII=A, link A2> @PE1,
adjacency-SID 2001 for <AII=A, link A1> @ASBR1, adjacency-SID 2002
for <AII=A, link A2> @ASBR2}.

For SR-TE instance 9 with color template which defined criteria
including {AII=B, min TE metric}, forwarding information could be
segment list {adjacency-SID 3004 for <AII=B, link B2> @PE1,
adjacency-SID 3003 for <AII=B, link B1> @ASBR1, adjacency-SID 3004
for <AII=B, link B2> @ASBR2}.

11.2.3. TE Service Using Flex-algo Example

For TE service, if we use SR Flex-algo to do optimization, the above
forwarding information of each TE instance could inherit the
corresponding SR FA entry, it would look like this:

For SR-TE instance 7, forwarding information could be ECMP during two
segment list {node-SID 600 for <FA-ID=201, destination=ASBR1> ,
adjacency-SID 1001 for <AII=0, link A1> @ASBR1, node-SID 600 for <FA-
ID=201, destination=PE2> } and {adjacency-SID 1004 for <AII=0, link
B2> @PE1, adjacency-SID 1003 for <AII=0, link B1> @ASBR1, adjacency-
SID 1004 for <AII=0, link B2> @ASBR2}.

For SR-TE instance 8 with color template which defined criteria
including {AII=A, min TE metric}, forwarding information could be
segment list {node-SID 700 for <FA-ID=202, destination=ASBR1> ,

adjacency-SID 2001 for <AII=A, link A1> @ASBR1, node-SID 700 for <FA-ID=202, destination=PE2> }.

For SR-TE instance 9 with color template which defined criteria including {AII=B, min TE metric}, forwarding information could be segment list {node-SID 800 for <FA-ID=203, destination=ASBR1> , adjacency-SID 3003 for <AII=B, link B1> @ASBR1, node-SID 800 for <FA-ID=203, destination=PE2> }.

11.3. Inter-domain Network Slicing via BGP-LU Example

In figure 1, PE2 can allocate and advertise six labels for its loopback plus color 1, 2, 3, 4, 5, 6 respectively. Suppose color 1 defines {default AII, min IGP metric}, color 2 defines {AII=A, min IGP metric}, color 3 defines {AII=B, min IGP metric}, and color 4 defines {default AII, min TE metric}, color 5 defines {AII=A, min TE metric}, color 6 defines {AII=B, min TE metric}. PE2 will advertise these labels to ASBR2 and ASBR2 then continues to allocate six labels each for prefix PE2 plus different color. Other nodes will have the same operation. Ultimately PE1 will maintain six BGP-LU LSP.

For example, the BGP-LU LSP for color 1 will be over SR best-effort FIB entry node-SID 100 for <AII=0, destination=ASBR1> to pass through AS1, over adjacency-SID 1001 for <AII=0, link A1>@ASBR1 to pass inter-AS, over SR best-effort FIB entry node-SID 400 for <AII=0, destination=PE2> to pass through AS2.

For example, The BGP-LU LSP for color 4 will over SR-TE instance 1 (see section 10.1.2), or SR best-effort FIB entry node-SID 600 for <FA-id=201, destination=ASBR1> (see section 10.1.3) to pass through AS1, over adjacency-SID 1001 for <AII=0, link A1>@ASBR1 to pass inter-AS, over SR-TE instance 1' or corresponding SR FA entry to pass through AS2. Note that ASBR1 need also understand the meaning of a specific color and select forwarding resource between two AS.

12. Implementation Suggestions

The implementation cost is low by means of existing segment routing infrastructure.

12.1. SR-MPLS

As a node often contains control plane and forwarding plane, a suggestion is that only default AII specific FTN table, i.e, traditional FTN table, need be installed on forwarding plane, so that there are not any modification and upgrade requirement for hardware and existing MPLS forwarding mechanism. FTN entry for non-default AII instance will only be maintained on the control plane and be used

for overlay service iteration according to next-hop plus color (color will give AII information and mapping FA-id information). Note that ILM entry for all AII need be installed on forwarding plane, that does not bring any confusion because of prefix-SID allocation per AII.

SR NHLFE entry and other iteration entry such as <next-hop, color> can contain AII information for expected packet scheduling. It is recommended that QoS policy per AII should be maintained in the underlay network. The Slice Type value of AII can distinguish flows by coarse-grained classification, while the Instance value of AII can be used for more scheduling policy.

12.2. SRv6

For SRv6 case, IPv6 address resource is directly used to represent SID, so that different IPv6 block could be allocated to different slice. There are two possible ways to advertise slice specific IPv6 block:

- o Traditional prefix reachability, but only for default AII (0) specific IPv6 block.
- o New SRv6 Locator advertisement, for nonzero AII specific IPv6 block.

Forwarding entries for the default AII specific locators advertised in prefix reachability MUST be installed in the forwarding plane of receiving routers.

Forwarding entries for the nonzero AII specific locators advertised in the SRv6 Locator MUST be also installed in the forwarding plane of receiving SRv6 capable routers when the associated AII is supported by the receiving node.

The entries of both the above two cases SHOULD be installed in the unified FIB table, i.e., a single FIB table for default AII, because different IPv6 block is allocated to different slice. Instead, more FIB tables created for each VN in dataplane will bring complexity for overlay service iteration, that is why MTR has no practical deployment.

The forwarding information of FIB entry can contain AII information for expected packet scheduling.

13. IANA Considerations

This document requests IANA to create a new top-level registry called "Network Slicing Parameters". This registry is being defined to serve as a top-level registry for keeping all other Network Slicing sub-registries.

Additionally, a new sub-registry "AII (TN-slice Identifier) codepoint" is to be created under top-level "Network Slicing Parameters" registry. This sub-registry maintains 32-bit identifiers and has the following registrations:

| Slice Type (High 8bits) | Instance (Low 24bits) | Description |
|----------------------------|--------------------------|--|
| 0 (Normal) | 0 | Reserved for Default Slice: the original physical network. |
| endogenous: IGP-metric | nonzero | Normal Slice, for user defined. |
| 1 (uRLLC) | 0 | Reserved. |
| endogenous: delay | nonzero | Slice suitable for the handling of ultra- reliable low latency communications, for user defined. |
| 2 (TE) | 0 | Reserved. |
| endogenous: TE-metric | nonzero | General TE Slice, for user defined. |
| 3 (eMBB) | 0 | Reserved. |
| endogenous: TBD | nonzero | Slice suitable for the handling of 5G enhanced Mobile Broadband, for user defined. |
| 4 (MIoT) | 0 | Reserved. |
| endogenous: TBD | nonzero | Slice suitable for the handling of massive IoT, for user defined. |
| 5 (V2X) | 0 | Reserved. |
| endogenous: TBD | nonzero | Slice suitable for the handling of V2X services, for user defined. |
| 6-255 | any | Unassigned. |

Table 1. AII Codepoint

14. Security Considerations

TBD.

15. Acknowledgements

TBD.

16. Normative references

- [I-D.ali-spring-network-slicing-building-blocks]
Ali, Z., Filsfils, C., Camarillo, P., and D. Voyer,
"Building blocks for Slicing in Segment Routing Network",
draft-ali-spring-network-slicing-building-blocks-02 (work
in progress), November 2019.
- [I-D.ietf-idr-bgppls-inter-as-topology-ext]
Wang, A., Chen, H., Talaulikar, K., and S. Zhuang, "BGP-LS
Extension for Inter-AS Topology Retrieval", draft-ietf-
idr-bgppls-inter-as-topology-ext-09 (work in progress),
September 2020.
- [I-D.ietf-idr-tunnel-encaps]
Patel, K., Velde, G., Sangli, S., and J. Scudder, "The BGP
Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-
encaps-19 (work in progress), September 2020.
- [I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and
A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-
algo-12 (work in progress), October 2020.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and
P. Mattes, "Segment Routing Policy Architecture", draft-
ietf-spring-segment-routing-policy-08 (work in progress),
July 2020.
- [I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca,
d., Li, C., Decraene, B., Ma, S., Yadlapalli, C.,
Henderickx, W., and S. Salsano, "Service Programming with
Segment Routing", draft-ietf-spring-sr-service-
programming-03 (work in progress), September 2020.
- [I-D.nsd-t-eas-transport-slice-definition]
Rokui, R., Homma, S., Makhijani, K., Contreras, L., and J.
Tantsura, "IETF Definition of Transport Slice", draft-
nsd-t-eas-transport-slice-definition-04 (work in
progress), September 2020.

- [I-D.peng-lsr-flex-algo-opt-slicing]
Peng, S., Chen, R., and G. Mirsky, "IGP Flexible Algorithm Optimazition for Netwrok Slicing", draft-peng-lsr-flex-algo-opt-slicing-02 (work in progress), September 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <<https://www.rfc-editor.org/info/rfc4915>>.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5329] Ishiguro, K., Manral, V., Davey, A., and A. Lindem, Ed., "Traffic Engineering Extensions to OSPF Version 3", RFC 5329, DOI 10.17487/RFC5329, September 2008, <<https://www.rfc-editor.org/info/rfc5329>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <<https://www.rfc-editor.org/info/rfc7308>>.

- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <<https://www.rfc-editor.org/info/rfc7810>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.
- [RFC8668] Ginsberg, L., Ed., Bashandy, A., Filsfils, C., Nanduri, M., and E. Aries, "Advertising Layer 2 Bundle Member Link Attributes in IS-IS", RFC 8668, DOI 10.17487/RFC8668, December 2019, <<https://www.rfc-editor.org/info/rfc8668>>.

Authors' Addresses

Shaofu Peng
ZTE Corporation

Email: peng.shaofu@zte.com.cn

Ran Chen
ZTE Corporation

Email: chen.ran@zte.com.cn

Gregory Mirsky
ZTE Corporation

Email: gregimirsky@gmail.com

Fengwei Qin
China Mobile

Email: qinfengwei@chinamobile.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

K. Raza, Ed.
R. Sawaya
Cisco Systems

Z. Shunwan
Huawei Technologies

D. Voyer
Bell Canada

M. Durrani
Equinix

S. Matsushima
SoftBank

V. Beeram
Juniper Networks

November 4, 2019

YANG Data Model for Segment Routing Policy
draft-raza-spring-sr-policy-yang-02.txt

Abstract

This document defines a YANG data model for Segment Routing (SR) Policy that can be used for configuring, instantiating, and managing SR policies. The model is generic and apply equally to the MPLS and SRv6 instantiations of SR policies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Specification of Requirements | 3 |
| 3. Building Blocks | 3 |
| 4. YANG Model | 4 |
| 4.1. Types and Definitions | 5 |
| 4.2. SR Policy | 6 |
| 4.2.1. Configuration | 6 |
| 4.2.2. State | 10 |
| 4.2.3. Notification | 12 |
| 5. Pending Items | 13 |
| 6. YANG Specification | 14 |
| 6.1. Types | 14 |
| 6.2. SR Policy | 22 |
| 7. Security Considerations | 44 |
| 8. IANA Considerations | 44 |
| 9. Acknowledgments | 45 |
| 10. References | 45 |
| 10.1. Normative References | 45 |
| 10.2. Informative References | 46 |
| Authors' Addresses | 46 |

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [RFC8402], allows a headend node to steer a packet flow along any topological path and/or service chain. The headend node is said to steer a flow into a Segment Routing Policy (SR Policy). An SR policy is a framework [I-D.ietf-spring-segment-routing-policy] that enables instantiation of an ordered list of segments on a node for implementing a policy.

This document introduces a YANG data model for SR policy framework for instantiating, configuring and managing SR policies along with its attributes. It is also expected that other companion models, such as BGP SR Policy [I-D.ietf-idr-segment-routing-te-policy], will be defined and/or augmented accordingly in their respective areas.

This model defines the following constructs for managing an SR policy:

- o Configuration
- o Operational State
- o Notifications
- o Executables (Actions)

This document expects and requires the reader to be well familiar with the concepts and constructs of an SR policy [I-D.ietf-spring-segment-routing-policy] as well as the YANG modeling language and its presentation [RFC6020].

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Building Blocks

Before looking into the YANG model for SR policy, it is important to recall and highlight the major building blocks and constructs that constitute and contribute to an SR policy, as described in [I-D.ietf-spring-segment-routing-policy].

- o **policy:** specifies constructs to allow a headend node to setup SR path(s) as an ordered list of segments for a given color and endpoint. The endpoint and the color are used to automate the steering of service or transport routes on an SR Policy. For a given headend, the key for an SR policy is (color, endpoint) where endpoint is an IP address that could be also NULL.
- o **candidate-path:** is the unit for signalling of an SR Policy to a headend via protocols (such as PCEP, BGP, CLI etc.). A candidate path is either dynamic or explicit type, where an explicit candidate path is associated with one or more segment-lists and

dynamic candidate path expresses optimization objectives and set of constraints. An SR Policy is associated with one or more candidate paths and the preference of the candidate path is used to select the best candidate path for an SR Policy. A candidate path is valid if it is usable (e.g. when its constituents SIDs are reachable). An "active" candidate path is the selected path (for forwarding) that is valid and determined to be the best path of the SR Policy. A candidate path is keyed by (protocol-origin, originator, discriminator).

- o segment-list: specifies ordered list of segments to traverse, where a segment can be specified in various forms (refer section 4 of [I-D.ietf-spring-segment-routing-policy]). The list is sorted by the index of the segment. A segment-list is used and referred by an explicit type of candidate-path. A segment-list is keyed by its name.
- o binding-sid: An SR policy is associated with a BSID to provide benefits of scaling, network opacity and service independence.

4. YANG Model

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

For SR policy YANG specification, this document defines following new YANG modules:

| Module Name | Purpose |
|----------------------|---|
| ietf-sr-policy-types | defines common and basic types related to an SR policy and related constructs |
| ietf-sr-policy | defines the model for SR policy instantiation, configuration, and management |

4.1. Types and Definitions

SR policy common types and definitions are defined in the new module "ietf-sr-policy-types". The main types defined in this module include:

- o dataplane-type: A union to specify MPLS or IPv6 as the dataplane type for SR.
- o sid-value-type: A Union to specify SID value for SR-MPLS or SRv6 type.
- o binding-sid-alloc-mode: Enum to define explicit or dynamic alloc mode types for a BSID.
- o protocol-origin-type: Enum to specify protocol origin (e.g. PCEP) for an SR policy.
- o explicit-binding-sid-rule-type: Enum to specify BSID alloc enforcement/rule when doing explicit alloc request.
- o binding-sid-oper-state: An Enum representing various operational states for a BSID.
- o policy-admin-state: An Enum for admin state of an SR policy.
- o policy-oper-state: An Enum for operational state of an SR policy.
- o segment-type: An Enum that defines various types for a "segment" of a Segment list.
- o candidate-path-non-selection-reason: The base identity along with its children to specify reason for not selecting a candidate path as the best/active path.
- o path-disjointness: The base identity for disjoint path computation. The disjointness types include link, node, srlg, srlg-node etc.
- o policy-down-reason: The base identity along with its children to specify reason for a policy becoming (or remaining) operationally down.
- o binding-sid-unavailable-reason: The base identity along with its children to specify reason for a BSID's unavailability.

The associated YANG specification for this module is captured in Section 6.1.

4.2. SR Policy

The base SR policy model is captured in `ietf-sr-policy` module. This base module augments `/rt:routing` and specifies the configuration, operational state, executables/rpcs, and notification events required to manage SR policies.

The associated YANG specification for this module is captured in Section 6.2.

4.2.1. Configuration

In terms of configuration hierarchy, SR policy configuration tree has following two main areas:

- o `attributes`: container that defines common constructs that could be used across policies. Examples of such a construct include `segment-lists`, `affinity-map` etc. In future revision of this document, it is expected that this container will have more constructs defined.
- o `policies`: container that defines list of policies with their attributes such as `BSID`, `candidate-paths` etc.

Following diagram depicts high level yang organization and hierarchy for an SR policy specification:

```

segment-routing
  traffic-engineering
    + attributes
    |
    | + affinity-map
    | | ....
    | |
    | + segment-lists
    | |   segment-list* [name]
    | |     segments
    | |       segment* [index]
    | |       ...
    | + explicit-binding-sid-rules
    |   ...
    + policies
      policy* [color endpoint]
      + ...
      |
      + binding-sid
      |   ...
      + candidate-paths
        candidate-path* [protocol-origin originator discriminato
r]
          + ...
          |
          + type
            + explicit
            |   segment-lists
            |     segment-list* [ref]
            |     ...
            + dynamic
              constraints
              ...

```

Figure 1: SR Policy - Hierarchy

Using the building blocks described in Section 3, following is the complete graphical representation of the data model for SR policy configuration:

```

module: ietf-sr-policy
  augment /rt:routing:
    +--rw segment-routing
    +--rw traffic-engineering
    +--rw attributes
    |   +--rw affinity-map

```

```

+--rw affinity* [name]
+--rw name string
+--rw bit-position? uint16
+--rw segment-lists
+--rw segment-list* [name]
+--rw name string
+--rw segments
+--rw segment* [index]
+--rw index uint32
+--rw type? sr-policy-types:segment-type
+--rw segment-types
+--rw segment-type-1
| +--rw sid-value? rt-types:mpls-label
+--rw segment-type-2
| +--rw sid-value? srv6-types:srv6-sid
+--rw segment-type-3
| +--rw ipv4-address? inet:ipv4-address
| +--rw algorithm? uint8
+--rw segment-type-4
| +--rw ipv6-address? inet:ipv6-address
| +--rw algorithm? uint8
+--rw segment-type-5
| +--rw ipv4-address? inet:ipv4-address
| +--rw interface-identifier? uint32
+--rw segment-type-6
| +--rw local-ipv4-address? inet:ipv4-address
| +--rw remote-ipv4-address? inet:ipv4-address
+--rw segment-type-7
| +--rw local-ipv6-address? inet:ipv6-ad
dress
| +--rw local-interface-identifier? uint32
| +--rw remote-ipv6-address? inet:ipv6-ad
dress
| +--rw remote-interface-identifier? uint32
+--rw segment-type-8
| +--rw local-ipv6-address? inet:ipv6-address
| +--rw remote-ipv6-address? inet:ipv6-address
+--rw segment-type-9
| +--rw ipv6-address? inet:ipv6-address
| +--rw algorithm? uint8
+--rw segment-type-10
| +--rw local-ipv6-address? inet:ipv6-ad
dress
| +--rw local-interface-identifier? uint32
| +--rw remote-ipv6-address? inet:ipv6-ad
dress
| +--rw remote-interface-identifier? uint32
+--rw segment-type-11
| +--rw local-ipv6-address? inet:ipv6-address
| +--rw remote-ipv6-address? inet:ipv6-address
+--rw validate? boolean
+--rw explicit-binding-sid-rules* [index]

```

```

|      +--rw index      uint32
|      +--rw rule?      sr-policy-types:explicit-binding-sid-rule-type
+--rw policies
  +--rw policy* [color endpoint]
    +--rw color          uint32
    +--rw endpoint       inet:ip-address
    +--rw name?          string
    +--rw description?   string
    +--rw admin-state?   sr-policy-types:policy-admin-state
    +--rw priority?      uint8
    +--rw binding-sid
      | +--rw dataplane?   sr-policy-types:dataplane-type
      | +--rw value?      sr-policy-types:sid-value-type
    +--rw candidate-paths
      +--rw candidate-path* [protocol-origin originator discriminat
or]
                                +--rw protocol-origin      sr-policy-types:protocol-o
rigin-type
                                +--rw originator            string
                                +--rw discriminator          uint32
                                +--rw preference             uint32
                                +--rw name?                  string
                                +--rw description?           string
                                +--rw binding-sid {capability-candidate-path-binding-sid}?
                                | +--rw dataplane?          sr-policy-types:dataplane-type
                                | +--rw value?              sr-policy-types:sid-value-type
                                +--rw (type)?
                                +--:(explicit)
                                | +--rw segment-lists
                                |   +--rw segment-list* [name-ref]
                                |     +--rw name-ref        -> /rt:routing/sr-policy:seg
ment-routing/traffic-engineering/attributes/segment-lists/segment-list/name
                                |     +--rw weight?         uint32
                                +--:(dynamic)
                                +--rw sid-dataplane-type?   sr-policy-types:data
plane-type
                                +--rw constraints
                                +--rw affinities
                                | +--rw exclude-any*       string
                                | +--rw include-any*        string
                                | +--rw include-all*       string
                                +--rw bounds
                                | +--rw igp-metric-bound?   uint32
                                | +--rw te-metric-bound?    uint32
                                | +--rw latency-metric-bound? uint32
                                | +--rw segment-bound?      uint32
                                +--rw segment-rules
                                | +--rw sid-algorithm?     uint8
                                +--rw disjoint-path
                                | +--rw group-id?          uint32
                                | +--rw disjointness-type?  identityref
                                | +--rw subgroup-id?       uint32

```


Figure 2: SR Policy - Config Tree

Please take note of the following important points in the above configuration model:

- o This model supports both MPLS and SRv6 dataplane for SR -- i.e. items like segments and BSID can be defined as MPLS label or SRv6 SIDs.
- o Specification of a segment supports all the types defined in SR policy base specification document
- o The above model supports explicit BSID specification on SR policy level as the main mode of specification. The model also allows explicit BSID per candidate-path as an if-feature capability that is optional for implementations
- o The above model will be extended in future revisions of this document to enhance constraints specification for dynamic type of candidate-path, as well as add traffic-steering controls.

4.2.2. State

As per NMDA model, the state related to configuration items specified in earlier Section 4.2.1 can be retrieved from the same tree. This section defines the other operational state items related to SR policy.

In addition to configured state, the operational state corresponding to the SR policy includes:

- o policy operational state
- o policy up/down timestamps
- o policy BSID info such as alloc mode, actual value in-use, operational state, and forwarding stats
- o Per candidate-path info such as:
 - * Whether candidate-path is the best candidate-path
 - * In case of non-best, the reason for such non-selection
 - * Type of candidate-path - explicit or dynamic
 - * Per segment-list information - such as validity of the segment-list, as well as forwarding state for a valid segment-list.

The forwarding state is represented in terms of per forwarding path info that includes nexthop address, outgoing interface, protection information, and encapsulation (label stack or SRv6 SID stack) etc.

Following is a simplified graphical representation of the data model for the SR policy (derived) operational state:

```

module: ietf-sr-policy
augment /rt:routing:
  +--rw segment-routing
  +--rw traffic-engineering
  +--rw policies
    +--rw policy* [color endpoint]
      +--rw color                uint32
      +--rw endpoint             inet:ip-address
      +--ro oper-state?         sr-policy-types:policy-oper-state
      +--ro transition-count?   uint32
      +--ro up-time?            yang:date-and-time
      +--ro down-time?          yang:date-and-time
      +--rw binding-sid
        +--ro alloc-mode?       sr-policy-types:binding-sid-alloc-mode
        +--ro allocated-sid?    sr-policy-types:sid-value-type
        +--ro oper-state?       sr-policy-types:binding-sid-oper-state
      +--ro counters
        +--ro pkts?             yang:counter64
        +--ro octets?           yang:counter64
      +--rw candidate-paths
        +--rw candidate-path* [protocol-origin originator discriminat
or]
          +--rw protocol-origin      sr-policy-types:protocol-o
rigin-type
          +--rw originator            string
          +--rw discriminator         uint32
          +--rw name                  string
          +--ro is-best-candidate-path? boolean
          +--ro non-selection-reason? identityref
          +--ro is-valid?             boolean
          +--ro forwarding-paths
            +--ro forwarding-path* [path-id]
              +--ro path-id          uint8
              +--ro next-hop-address? inet:ip-address
              +--ro next-hop-table-id? uint32
              +--ro interface?       if:interface-ref
              +--ro sid-list
                +--ro (dataplanetype)?
                  +--:(mpls)
                    +--ro labels* [label]
                      +--ro label    rt-types:mpls-label

```

```

      |      +--:(srv6)
      |      |      +--ro sids* [sid]
      |      |      |      +--ro sid      srv6-types:srv6-sid
+--ro is-protected?      boolean
+--ro is-pure-backup?    boolean
+--ro backup-path-id?    uint8
+--ro weight?            uint32

```

Figure 3: SR Policy - State Tree

4.2.3. Notification

This model defines a list of notifications to inform an operator of important events detected regarding an SR policy. These events include events related to:

- o policy status: policy operational state changes
- o Candidate-path active status and changes
- o Explicit Binding SID collision/unavailability events

Following is a simplified graphical representation of the data model for SR policy notifications:

```
module: ietf-sr-policy
```

```

  notifications:
    +---n sr-policy-oper-state-change-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-routing/t
raffic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-routing/t
raffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-routing/t
raffic-engineering/policies/policy/endpoint
    |   +---ro policy-new-oper-state?    sr-policy-types:policy-oper-state
    |   +---ro policy-down-reason?      identityref
    +---n sr-policy-candidate-path-change-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-routing/tra
ffic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-routing/tra
ffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-routing/tra
ffic-engineering/policies/policy/endpoint
    |   +---ro existing-preference?      uint32
    |   +---ro new-preference?           uint32
    +---n sr-policy-binding-sid-unavailable-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-ro
uting/traffic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-ro
uting/traffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-ro
uting/traffic-engineering/policies/policy/endpoint
    |   +---ro policy-binding-sid-value-ref? -> /rt:routing/sr-policy:segment-ro
uting/traffic-engineering/policies/policy/binding-sid/value
    |   +---ro reason?                  identityref
    +---n sr-policy-candidate-path-binding-sid-mismatch-event
    |   +---ro policy-color-ref?          -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?       -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/endpoint
    |   +---ro existing-candidate-path-protocol-origin-ref? -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/c
andidate-path/protocol-origin
    |   +---ro existing-candidate-path-preference-ref?       -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/c
andidate-path/preference
    |   +---ro existing-candidate-path-binding-sid-dataplane-ref? -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/c
andidate-path/binding-sid/dataplane
    |   +---ro existing-candidate-path-binding-sid-value-ref? -> /rt:routing
/sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/c
andidate-path/binding-sid/value
    |   +---ro conflicting-candidate-path-protocol-origin?    uint8
    |   +---ro conflicting-candidate-path-preference?         uint32
    |   +---ro conflicting-candidate-path-binding-sid-dataplane? sr-policy-type
s:dataplane-type
    |   +---ro conflicting-candidate-path-binding-sid-value?  sr-policy-type
s:sid-value-type

```

Figure 4: SR Policy - Notification Tree

5. Pending Items

Following are the items that will be addressed in future revisions of this document:

- o Configuration and Specification of:

- * Traffic steering over SR policy

- * ODN templates

- * Spray policy

- o Executables (RPC actions)

6. YANG Specification

Following are actual YANG definition for the modules defined earlier in the document.

6.1. Types

```
<CODE BEGINS> file "ietf-sr-policy-types@2019-11-04.yang" -->

module ietf-sr-policy-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy-types";

  prefix "sr-policy-types";

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  import ietf-srv6-types {
    prefix "srv6-types";
  }

  organization "IETF SPRING Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/spring/>
    WG List:    <mailto:spring@ietf.org>
    Editor:     Kamran Raza
                <mailto:skraza@cisco.com>
    Editor:     Zhuang Shunwan
                <mailto:zhuangshunwa@huawei.com>
    Editor:     Daniel Voyer
                <mailto:daniel.voyer@bell.ca>
    Editor:     Muhammad Durrani
                <mailto:mdurrani@equinix.com>
    Editor:     Satoru Matsushima
                <mailto:satoru.matsushima@g.softbank.co.jp>
    Editor:     Pavan Vishnu Beeram
                <mailto:vbeeram@juniper.net>
    ";
```

```
description
  "This YANG module defines the essential types for the management
  of SR policy module.
  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.
  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).";

revision "2019-11-04" {
  description
    "New editor added";
  reference
    "draft-raza-spring-sr-policy-yang";
}

revision "2019-07-08" {
  description
    "Dynamic TE candidate-path support";
  reference
    "draft-raza-spring-sr-policy-yang";
}

revision "2018-07-01" {
  description
    "Initial version";
  reference
    "draft-raza-spring-sr-policy-yang";
}

/* Identities */
identity candidate-path-not-selected-reason {
  description
    "Base identity for which reasons for not selecting
    candidate path are derived from";
}
identity candidate-path-not-selected-not-best {
  base candidate-path-not-selected-reason;
  description
    "Higher preference path exists";
}
identity candidate-path-not-selected-no-valid-segment-list {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has no valid segment list(s)";
}
```

```
}
identity candidate-path-not-selected-empty-segment-list {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has empty segment list(s)";
}
identity candidate-path-not-selected-invalid-binding-sid {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has invalid binding SID";
}

identity policy-down-reason {
  description
    "Base identity for the reasons why SR policy is operationally down";
}
identity policy-down-reason-admin-down {
  base policy-down-reason;
  description "Policy is administrately down";
}
identity policy-down-reason-no-source-address {
  base policy-down-reason;
  description "Policy has no source address";
}
identity policy-down-reason-no-endpoint {
  base policy-down-reason;
  description "Policy has no end-point";
}
identity policy-down-reason-no-candidate-path {
  base policy-down-reason;
  description "Policy has no candidate path";
}
identity policy-down-reason-no-valid-candidate-path {
  base policy-down-reason;
  description "Policy has no valid candidate path";
}
identity policy-down-reason-candidate-path-invalid-segment-list {
  base policy-down-reason;
  description "Policy's candidate path has invalid segment list";
}
identity policy-down-reason-policy-unconfigured {
  base policy-down-reason;
  description "Policy is unconfigured";
}
identity policy-down-reason-policy-color-endpoint-updated {
  base policy-down-reason;
  description "Policy's color and end-point are updated";
}
```



```
identity policy-down-reason-local-label-setup-failed {
  base policy-down-reason;
  description "Policy's local label setup (allocation/rewrite) failed";
}
identity policy-down-reason-forwarding-rewrite-failed {
  base policy-down-reason;
  description "Policy's forwarding rewrite installation failed";
}
identity policy-down-reason-internal-error {
  base policy-down-reason;
  description "Infra related internal error";
}

identity binding-sid-unavailable-reason {
  description
    "Base identity for binding sid unavailable reason types";
}
identity binding-sid-allocation-error {
  base binding-sid-unavailable-reason;
  description "SID allocator returned an error";
}
identity binding-sid-already-exists {
  base binding-sid-unavailable-reason;
  description "Binding sid already exists/allocated";
}
identity binding-sid-internal-error {
  base binding-sid-unavailable-reason;
  description "Internal error with binding sid allocation";
}
identity binding-sid-color-endpoint-conflict {
  base binding-sid-unavailable-reason;
  description "Binding sid already allocated by another sr-policy with different color/endpoint";
}
identity binding-sid-rewrite-error {
  base binding-sid-unavailable-reason;
  description "Binding sid forwarding rewrite error";
}
identity binding-sid-outside-srlb-range {
  base binding-sid-unavailable-reason;
  description "Binding sid outside SRLB range";
}

identity path-disjointness {
  description
    "Base identity for the type of path disjointness computation";
}
identity path-disjointness-link {
  base path-disjointness;
```

```
        description "The computed path is link-disjoint with the existing path";
    }
    identity path-disjointness-node {
        base path-disjointness;
        description "The computed path is node-disjoint with the existing path";
    }
    identity path-disjointness-srlg {
        base path-disjointness;
        description "The computed path is srlg-disjoint with the existing path";
    }
    identity path-disjointness-srlg-node {
        base path-disjointness;
        description "The computed path is node and srlg disjoint with the existing p
ath";
    }

    /* Typdefs */
    typedef sid-value-type {
        type union {
            type rt-types:mpls-label;
            type srv6-types:srv6-sid;
        }
        description "The SID value type";
    }

    typedef binding-sid-oper-state {
        type enumeration {
            enum ALLOC-PENDING {
                value 1;
                description "SID allocation pending for Binding SID";
            }
            enum PROGRAMMED {
                value 3;
                description "Binding SID is programmed in forwarding";
            }
            enum CONFLICT {
                value 4;
                description "Binding SID is in-conflict state with
                regards to SID allocation. This also means that SID
                allocation is pending";
            }
        }
        description
            "Binding SID operational state type";
    }

    typedef policy-admin-state {
        type enumeration {
            enum UP {
```

```
        value 1;
        description "SR policy is administratively up";
    }
    enum DOWN {
        value 2;
        description "SR policy is administratively down";
    }
}
description "SR policy admin state";
}

typedef policy-oper-state {
    type enumeration {
        enum UP {
            value 1;
            description "SR policy is operationally up";
        }
        enum DOWN {
            value 2;
            description "SR policy is operationally down";
        }
    }
}
description "SR policy oper state";
}

typedef segment-type {
    type enumeration {
        enum segment-type-1 {
            value 1;
            description "SR-MPLS Label";
        }
        enum segment-type-2 {
            value 2;
            description "SRv6 SID";
        }
        enum segment-type-3 {
            value 3;
            description "IPv4 Prefix with optional SR Algorithm";
        }
        enum segment-type-4 {
            value 4;
            description "IPv6 Global Prefix with optional SR Algorithm for SR-MPLS";
        }
        enum segment-type-5 {
            value 5;
            description "IPv4 Prefix with Local Interface ID";
        }
        enum segment-type-6 {
```

```
        value 6;
        description "IPv4 Addresses for link endpoints as Local, Remote pair";
    }
    enum segment-type-7 {
        value 7;
        description "IPv6 Prefix and Interface ID for link endpoints as Local,
            Remote pair for SR-MPLS";
    }
    enum segment-type-8 {
        value 8;
        description "IPv6 Addresses for link endpoints as Local, Remote pair for
            SR-MPLS";
    }
    enum segment-type-9 {
        value 9;
        description "IPv6 Global Prefix with optional SR Algorithm for SRv6";
    }
    enum segment-type-10 {
        value 10;
        description "IPv6 Prefix and Interface ID for link endpoints as Local,
            Remote pair for SRv6";
    }
    enum segment-type-11 {
        value 11;
        description "IPv6 Addresses for link endpoints as Local, Remote pair for
            SRv6";
    }
}
description "SR segment type";
}

typedef dataplane-type {
    type enumeration {
        enum mpls {
            value 1;
            description "Segment-routing MPLS";
        }
        enum srv6 {
            value 2;
            description "Segment-routing v6";
        }
    }
    description "Dataplane type of the segments";
}

typedef binding-sid-alloc-mode {
    type enumeration {
        enum explicit {
```

```
        value 1;
        description "Explicitly specified BSID";
    }
    enum dynamic {
        value 2;
        description "Dynamically allocated BSID";
    }
}
description "binding SID allocation mode";
}

typedef protocol-origin-type {
    type enumeration {
        enum pcep {
            value 10;
            description "PCEP used as signalling mechanism for the candidate path";
        }
        enum bgp {
            value 20;
            description "BGP used as signalling mechanism for the candidate path";
        }
        enum local {
            value 30;
            description "CLI, Yang model via Netconf, gRPC, etc used for candidate path instantiation";
        }
    }

    description "Originating protocol type";
}

typedef explicit-binding-sid-rule-type {
    type enumeration {
        enum enforce-srlb {
            value 1;
            description
                "Explicit Binding SID is enforced with no
                 fallback if label does not fall in SRLB or
                 if no SRLB is configured";
        }
        enum fallback-dynamic {
            value 2;
            description
                "Explicit Binding SID falls back to dynamic in
                 case explicit label is not available.";
        }
    }
    description "Explicit binding sid rule types";
}
```

```
} // module
```

```
<CODE ENDS>
```

Figure 5: ietf-sr-policy-types.yang

6.2. SR Policy

```
<CODE BEGINS> file "ietf-sr-policy@2019-11-04.yang" -->
```

```
module ietf-sr-policy {  
    namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy";  
  
    prefix "sr-policy";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ietf-interfaces {  
        prefix if;  
    }  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-routing-types {  
        prefix "rt-types";  
    }  
  
    import ietf-yang-types {  
        prefix "yang";  
    }  
  
    import ietf-srv6-types {  
        prefix "srv6-types";  
    }  
  
    import ietf-sr-policy-types {  
        prefix "sr-policy-types";  
    }  
  
    organization "IETF SPRING Working Group";
```

contact

"WG Web: <<http://tools.ietf.org/wg/spring/>>
WG List: <<mailto:spring@ietf.org>>

Editor: Kamran Raza
<<mailto:skraza@cisco.com>>

Editor: Zhuang Shunwan
<<mailto:zhuangshunwa@huawei.com>>

Editor: Daniel Voyer
<<mailto:daniel.voyer@bell.ca>>

Editor: Muhammad Durrani
<<mailto:mdurrani@equinix.com>>

Editor: Satoru Matsushima
<<mailto:satoru.matsushima@g.softbank.co.jp>>

Editor: Pavan Vishnu Beeram
<<mailto:vbeeram@juniper.net>>
";

description

"This module contains a collection of YANG definitions
for SR policy module.

Copyright (c) 2019 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).";

```
revision "2019-11-04" {  
  description  
    "Changes in keys for policy and its candidate paths";  
  reference  
    "draft-raza-spring-sr-policy-yang";  
}
```

```
revision "2019-07-08" {  
  description  
    "Dynamic TE candidate-path support";
```

```
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  revision "2018-07-01" {
    description
      "Initial version";
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  grouping segment_config {
    description "Segment properties grouping";
    leaf index {
      type uint32;
      description "Segment index";
    }
    leaf type {
      type sr-policy-types:segment-type;
      description "Segment type";
    }
    container segment-types {
      description "Types of segments";
      container segment-type-1 {
        description
          "Segment declared by MPLS label";
        leaf sid-value {
          type rt-types:mpls-label;
          description "MPLS label value";
        }
      }
      container segment-type-2 {
        description
          "Segment declared by SRv6 SID value";
        leaf sid-value {
          type srv6-types:srv6-sid;
          description "SRv6 SID value";
        }
      }
      container segment-type-3 {
        description
          "Segment declared by IPv4 Prefix with optional SR Algorithm";
        leaf ipv4-address {
          type inet:ipv4-address;
          description "Segment IPv4 address";
        }
        leaf algorithm {
          type uint8;
        }
      }
    }
  }
```



```
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-4 {
    description
        "Segment declared by IPv6 Global Prefix with optional
        SR Algorithm for SR-MPLS";
    leaf ipv6-address {
        type inet:ipv6-address;
        description "Segment IPv6 address";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-5 {
    description
        "Segment declared by IPv4 Prefix with Local Interface ID";
    leaf ipv4-address {
        type inet:ipv4-address;
        description "Node IPv4 address";
    }
    leaf interface-identifier {
        type uint32;
        description "local interface identifier";
    }
}
container segment-type-6 {
    description
        "Segment declared by IPv4 Addresses for link endpoints
        as Local, Remote pair";
    leaf local-ipv4-address {
        type inet:ipv4-address;
        description "Segment local IPv4 adjacency address";
    }
    leaf remote-ipv4-address {
        type inet:ipv4-address;
        description "Segment remote IPv4 adjacency address";
    }
}
container segment-type-7 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for
        link endpoints as Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Local link IPv6 address";
    }
}
```

```
    }
    leaf local-interface-identifier {
        type uint32;
        description "Local interface identifier";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Remote link IPv6 address";
    }
    leaf remote-interface-identifier {
        type uint32;
        description "Remote interface identifier";
    }
}
container segment-type-8 {
    description
        "Segment declared by IPv6 Addresses for link endpoints as
        Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Segment remote IPv6 adjacency address";
    }
}
container segment-type-9 {
    description
        "Segment declared by IPv6 Global Prefix with optional
        SR Algorithm for SRv6";
    leaf ipv6-address {
        type inet:ipv6-address;
        description "Segment IPv6 prefix";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-10 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for
        link endpoints as Local, Remote pair for SRv6";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Local link IPv6 address";
    }
}
```

```

    leaf local-interface-identifier {
        type uint32;
        description "Local interface identifier";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Remote link IPv6 address";
    }
    leaf remote-interface-identifier {
        type uint32;
        description "Remote interface identifier";
    }
}
container segment-type-11 {
    description
        "Segment declared by IPv6 Addresses for link endpoints as
        Local, Remote pair for SRv6";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Segment remote IPv6 adjacency address";
    }
}
}
leaf validate {
    type boolean;
    default 'false';
    description "Indicates whether the segment should be validated. The default
t
        applies to all segments other than the first segment. For the
        first segment, validation is always done.";
}
}

grouping segment-properties {
    description
        "SR segment properties grouping";
    uses segment_config;
}

grouping attributes {
    description
        "Grouping containing attributes applicable to all SR policies";

    container attributes {
        description

```

```
        "Attributes applicable to SR policies";

        uses affinity-mapping;
        uses segment-lists;
        uses explicit-binding-sid-rules;
    }
}

grouping segment-lists {
    description
        "Segment lists grouping";
    container segment-lists {
        description "Segment-lists properties";

        list segment-list {
            key "name";
            description "Segment-list properties";
            leaf name {
                type string;
                description "Segment-list name";
            }
            container segments {
                description
                    "Segments for given segment list";

                list segment {
                    key "index";
                    description "Configure Segment/hop at the index";
                    uses segment-properties;
                }
            }
        }
    }
}

grouping binding-sid_config {
    description
        "Binding SID configuration properties grouping";
    leaf dataplane {
        type sr-policy-types:dataplane-type;
        description "Binding SID dataplane type";
    }
    leaf value {
        type sr-policy-types:sid-value-type;
        description "Binding SID value";
    }
}
```

```
grouping forwarding-counters {
  description
    "Grouping for counters";
  container counters {
    config false;
    description
      "Counters containing stats related to forwarding";

    leaf pkts {
      type yang:counter64;
      description "Number of packets forwarded";
    }
    leaf octets {
      type yang:counter64;
      units "byte";
      description "Number of bytes forwarded";
    }
  }
}

grouping binding-sid_state {
  description
    "Binding SID state properties grouping";
  leaf alloc-mode {
    type sr-policy-types:binding-sid-alloc-mode;
    config false;
    description "Binding SID type";
  }
  leaf allocated-sid {
    type sr-policy-types:sid-value-type;
    config false;
    description "Allocated SID value for the Binding SID";
  }
  leaf oper-state {
    type sr-policy-types:binding-sid-oper-state;
    config false;
    description
      "Binding SID operational state";
  }
}

grouping binding-sid-properties {
  description
    "Binding SID properties grouping";
  container binding-sid {
    description "Binding Segment ID";
    uses binding-sid_config;
    uses binding-sid_state;
  }
}
```

```
    }
  }

  grouping mpls-label-stack {
    description
      "Grouping for MPLS label stack";

    list labels {
      key "label";
      description
        "Stack containing MPLS labels";

      leaf label {
        type rt-types:mpls-label;
        description
          "MPLS label value";
      }
    }
  }

  grouping srv6-sid-stack {
    description
      "Grouping for SRv6 label stack";

    list sids {
      key "sid";
      description
        "Stack containing SRv6 SIDs";

      leaf sid {
        type srv6-types:srv6-sid;
        description
          "SRv6 sid value";
      }
    }
  }

  grouping path-forwarding_state {
    description "Policy Forwarding path information";
    leaf path-id {
      type uint8;
      description "Primary path id";
    }
    leaf next-hop-address {
      type inet:ip-address;
      description "Nexthop address";
    }
    leaf next-hop-table-id {
```

```
    type uint32;
    description "Table ID for nexthop address";
  }
  leaf interface {
    type if:interface-ref;
    description "Outgoing interface handle";
  }
  container sid-list {
    description
      "Outgoing sid stack";
    choice dataplanetype {
      description
        "Outgoing sids dataplane choice";
      case mpls {
        uses mpls-label-stack;
      }
      case srv6 {
        uses srv6-sid-stack;
      }
    }
  }
}
leaf is-protected {
  type boolean;
  description "Is this path protected ?";
}
leaf is-pure-backup {
  type boolean;
  description "Is this path a pure backup ?";
}
leaf backup-path-id {
  type uint8;
  description "Backup path id";
}
leaf weight {
  type uint32;
  description "Path's weight for W-ECMP balancing";
}
}

grouping cpath-cmn-properties {
  description
    "Common properties of the candidate path";

  leaf is-valid {
    type boolean;
    config false;
    description
      "True if the segment-list is valid, False otherwise";
  }
}
```

```
    }

    container forwarding-paths {
        config false;
        description
            "Forwarding state of paths";
        list forwarding-path {
            key "path-id";
            description "Forwarding path";
            uses path-forwarding_state;
        }
    }
}

grouping explicit-path-properties {
    description
        "Explicit path properties of the candidate path";
    container segment-lists {
        description
            "Path segment list(s) properties";
        list segment-list {
            key "name-ref";
            description "SR policy candidate path segment lists";

            leaf name-ref {
                type leafref {
                    path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engine
ering/sr-policy:attributes/sr-policy:segment-lists/sr-policy:segment-list/sr-pol
icy:name";
                }
                description "Reference to segment-list name";
            }
            leaf weight {
                type uint32;
                description "Segment-list weighted loadshare";
            }
        }
    }
}

grouping affinity-mapping {
    description "Affinity-map grouping";

    container affinity-map {
        description
            "Mapping of affinity names to bit position";
        list affinity {
            key "name";
            unique "bit-position";
            leaf name {
```



```
        type string;
        description
            "Name of the affinity";
    }
    leaf bit-position {
        type uint16;
        description
            "The affinity entry in this list is mapped to the this bit-position
in the
        affinity bitmap";
    }

    description "Affinity";
}
}
}
grouping dynamic-path-properties {
    description
        "Dynamic path properties of the candidate path";
    leaf sid-dataplane-type {
        type sr-policy-types:dataplane-type;
        description
            "The dataplane type for the sid";
    }
}

container constraints {
    description "Constraints for the dynamic path computation";
    container affinities {
        description "Affinity constraints on the computed dynamic path";
        leaf-list exclude-any {
            type string;
            description
                "The link is excluded if it has any of these affinities.";
        }
        leaf-list include-any {
            type string;
            description
                "The link is accepted if it has any of these affinities";
        }
        leaf-list include-all {
            type string;
            description
                "The link is accepted if it has all these affinities";
        }
    }
}

container bounds {
    description "Upper-bound constraints on the computed dynamic path";
    leaf igp-metric-bound {
```

```
        type uint32;
        description
            "Path is invalid if its IGP metric exceeds this value";
    }
    leaf te-metric-bound {
        type uint32;
        description
            "Path is invalid if its TE metric exceeds this value";
    }
    leaf latency-metric-bound {
        type uint32;
        units "microsecond";
        description
            "Path is invalid if its latency exceeds this value";
    }
    leaf segment-bound {
        type uint32;
        description
            "Path is invalid if it has more segments than this value";
    }
}
container segment-rules {
    description "Constraints on the segments to be used in the path";
    leaf sid-algorithm {
        type uint8 {
            range "128..255";
        }
        description
            "The prefix-sid algorithm to be used in path calculation";
    }
}
container disjoint-path {
    description "Path disjointness constraints";
    leaf group-id {
        type uint32 { range "1..65535"; }
        description "";
    }
    leaf disjointness-type {
        type identityref { base sr-policy-types:path-disjointness; }
        description
            "Type of disjointness computation used to find the path";
    }
    leaf subgroup-id {
        type uint32 { range "1..65535"; }
        description "";
    }
}
}
```

```
    }

    grouping candidate-path_state {
      description
        "Candidate path state properties grouping";
      leaf is-best-candidate-path {
        type boolean;
        default 'false';
        config false;
        description
          "True if the candidate path is the best candidate path, False otherwise"
      };
    }
    leaf non-selection-reason {
      type identityref {
        base sr-policy-types:candidate-path-not-selected-reason;
      }
      config false;
      description
        "Candidate path not selected reason";
    }
  }

  grouping policy-properties_config {
    description
      "SR policy configuration grouping";
    leaf name {
      type string {
        length "1..59";
      }
      description "SR policy name";
    }
    leaf color {
      type uint32 {
        range "1..4294967295";
      }
      description "Color associated with the policy";
    }
    leaf endpoint {
      type inet:ip-address;
      description "Policy end point IP address";
    }
    leaf description {
      type string;
      description "Description of the policy";
    }
    leaf admin-state {
      type sr-policy-types:policy-admin-state;
      default 'UP';
    }
  }
}
```

```
        description
          "SR policy administrative state, true for
           enabled, false for disabled";
      }
    }

    grouping policy-properties_state {
      description
        "SR policy property grouping";
      leaf oper-state {
        type sr-policy-types:policy-oper-state;
        config false;
        description
          "SR policy operational state";
      }
      leaf transition-count {
        type uint32;
        config false;
        description "Indicates number of up/down transitions";
      }
      leaf up-time {
        type yang:date-and-time;
        config false;
        description "Policy up time in seconds";
      }
      leaf down-time {
        type yang:date-and-time;
        config false;
        description "Policy down time in seconds";
      }
    }

    grouping policy-properties {
      description
        "SR policy properties";
      uses policy-properties_state;
      uses binding-sid-properties;
      uses forwarding-counters;
    }

    grouping candidate-path-type {
      description "Candidate path type grouping";
      choice type {
        description
          "Type of candidate paths";
        case explicit {
          description "Candidate path with explicitly defined set/s of segment-lis
ts";
          uses explicit-path-properties;
        }
      }
    }
  }
}
```

```
    }
    case dynamic {
      description "Candidate path with dynamic computed segment-lists";
      uses dynamic-path-properties;
    }
  }
}

grouping candidate-paths {
  description "SR policy candidate path grouping";
  container candidate-paths {
    description "SR policy candidate path(s) ";

    list candidate-path {
      key "protocol-origin originator discriminator";
      unique "preference";

      description "SR policy Candidate path(s) list entry";

      leaf protocol-origin {
        type sr-policy-types:protocol-origin-type;
        description
          "Instantiation mechanism used to create the candidate path";
      }
      leaf originator {
        type string;
        description
          "Identifier (concatenation of ASN and node-address) of the node
           that signalled/instantiated the candidate path on headend";
      }
      leaf discriminator {
        type uint32;
        description "Candidate path distinguisher";
      }

      leaf preference {
        type uint32 {
          range "1..65535";
        }
        mandatory true;
        description "Candidate path preference";
      }
      leaf name {
        type string;
        description "Candidate path name";
      }
      leaf description {
        type string;
      }
    }
  }
}
```

```
        description "Candidate path description";
    }
    container binding-sid {
        if-feature capability-candidate-path-binding-sid;
        description
            "Binding segment ID";
        uses binding-sid_config;
    }

    uses candidate-path-type;
    uses candidate-path_state;
    uses cpath-cmn-properties;
}
}

grouping policies {
    description "SR policy grouping";
    container policies {
        description "SR Policy container";

        list policy {
            key "color endpoint";
            unique "name";

            description "SR Policy properties";
            leaf color {
                type uint32 {
                    range "1..4294967295";
                }
                description "Color associated with the policy";
            }
            leaf endpoint {
                type inet:ip-address;
                description "Policy end point IP address";
            }
            leaf name {
                type string {
                    length "1..59";
                }
                description "SR policy name";
            }
            leaf description {
                type string;
                description "Description of the policy";
            }
            leaf admin-state {
                type sr-policy-types:policy-admin-state;
            }
        }
    }
}
```

```
        default 'UP';
        description
            "SR policy administrative state, true for
            enabled, false for disabled";
    }
    leaf priority {
        type uint8;
        default 128;
        description "Priority considered when policy is recomputed due to topo
logy changes";
    }

    uses policy-properties;

    uses candidate-paths;
}
}

grouping explicit-binding-sid-rules {
    description
        "Grouping for explicit binding sid rules";

    list explicit-binding-sid-rules {
        key "index";
        description
            "Explicit binding sid rules applicable for all policies";
        leaf index {
            type uint32;
            description "Explicit binding SID rules list index";
        }
        leaf rule {
            type sr-policy-types:explicit-binding-sid-rule-type;
            description "Explicit binding sid rule";
        }
    }
}

augment "/rt:routing" {
    description
        "This augments routing-instance configuration with segment-routing sr-pol
icy.";
    container segment-routing {
        description "Main segment routing container";
        container traffic-engineering {
            description "Traffic-engineering container";

            uses attributes;

            uses policies;
        }
    }
}
```

```
    }
  }
}

/* Notifications */

notification sr-policy-oper-state-change-event {
  description
    "Notification event when the operational state of the SR policy changes";

  leaf policy-name-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:name";
    }
    description "Reference to sr-policy name";
  }

  leaf policy-color-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:color";
    }
    description "Reference to sr-policy color";
  }

  leaf policy-endpoint-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
    }
    description "Reference to sr-policy endpoint";
  }

  leaf policy-new-oper-state {
    type sr-policy-types:policy-oper-state;
    description "New operational state of the SR policy";
  }

  leaf policy-down-reason {
    type identityref {
      base sr-policy-types:policy-down-reason;
    }
    description "Down reason if the SR policy's new operational state is down"
;
  }
}

notification sr-policy-candidate-path-change-event {
  description
    "Notification event when candidate path changes for SR policy";
```



```
    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-preference {
      type uint32;
      description "Existing candidate path preference";
    }

    leaf new-preference {
      type uint32;
      description "New candidate path preference";
    }
  }

  notification sr-policy-binding-sid-unavailable-event {
    description
      "Notification event when the binding sid of sr-policy is unavailable";

    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }
  }
}
```

```
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf policy-binding-sid-value-ref {
      if-feature capability-candidate-path-binding-sid;
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:binding-sid/sr-policy:value";
      }
      description "Reference to sr-policy binding-sid value";
    }

    leaf reason {
      type identityref {
        base sr-policy-types:binding-sid-unavailable-reason;
      }
      description
        "Reason why the binding sid is unavailable";
    }
  }

  notification sr-policy-candidate-path-binding-sid-mismatch-event {
    description
      "Notification event when binding sid of requested candidate path
      is different from the binding sid of the existing candidate path";

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-candidate-path-protocol-origin-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering
/sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candida
te-path/sr-policy:protocol-origin";
      }
    }
  }
}
```

```
    description "Reference to existing candidate path protocol origin";
  }

  leaf existing-candidate-path-preference-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:preference";
    }
    description "Reference to existing candidate path preference";
  }

  leaf existing-candidate-path-binding-sid-dataplane-ref {
    if-feature capability-candidate-path-binding-sid;
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:binding-sid/sr-policy:dataplane";
    }
    description "Reference to existing candidate path binding sid dataplane type";
  }

  leaf existing-candidate-path-binding-sid-value-ref {
    if-feature capability-candidate-path-binding-sid;
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:binding-sid/sr-policy:value";
    }
    description "Reference to existing candidate path binding sid value";
  }

  leaf conflicting-candidate-path-protocol-origin {
    type uint8;
    description "Conflicting candidate path protocol origin";
  }

  leaf conflicting-candidate-path-preference {
    type uint32;
    description "Conflicting candidate path preference";
  }

  leaf conflicting-candidate-path-binding-sid-dataplane {
    type sr-policy-types:dataplane-type;
    description "Conflicting candidate path binding sid dataplane type";
  }

  leaf conflicting-candidate-path-binding-sid-value {
    type sr-policy-types:sid-value-type;
    description "Conflicting candidate path binding sid value";
  }
}

/* Features */

feature capability-candidate-path-binding-sid {
```

```

    description
      "This feature enables the capability of specifying binding-sid
       for a candidate path.";
  }
} // module

<CODE ENDS>

```

Figure 6: ietf-sr-policy.yang

7. Security Considerations

The configuration, state, and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC8341].

8. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

| URI | Registrant | XML |
|--|------------|-----|
| urn:ietf:params:xml:ns:yang:ietf-sr-policy-types | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-sr-policy | The IESG | N/A |

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

| Name | Namespace | Prefix | Reference |
|----------------------|--|-----------------|---------------|
| ietf-sr-policy-types | urn:ietf:params:xml:ns:yang:ietf-sr-policy-types | sr-policy-types | This document |
| ietf-sr-policy | urn:ietf:params:xml:ns:yang:ietf-sr-policy | sr-policy | This document |

9. Acknowledgments

The authors of this document/YANG model would like to acknowledge the contributions/reviews by Johnson Thomas, Clarence Filsfils, Siva Sivabalan, Tarek Saad, Kris Michielsen, Dhanendra Jain, Ketan Talaulikar, Bhupendra Yadav, and Bruno Decraene.

10. References

10.1. Normative References

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.

10.2. Informative References

- [I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Mattes, P., Rosen, E., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-07 (work in progress), July 2019.

Authors' Addresses

Kamran Raza (editor)
Cisco Systems
Email: skraza@cisco.com

Robert Sawaya
Cisco Systems
Email: rsawaya@cisco.com

Zhuang Shunwan
Huawei Technologies
Email: zhuangshunwa@huawei.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Muhammad Durrani
Equinix
Email: mdurrani@equinix.com

Satoru Matsushima
SoftBank
Email: satoru.matsushima@g.softbank.co.jp

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

K. Raza, Ed.
R. Sawaya
Cisco Systems

Z. Shunwan
Huawei Technologies

D. Voyer
Bell Canada

M. Durrani
Equinix

S. Matsushima
SoftBank

V. Beeram
Juniper Networks

July 13, 2020

YANG Data Model for Segment Routing Policy
draft-raza-spring-sr-policy-yang-03.txt

Abstract

This document defines a YANG data model for Segment Routing (SR) Policy that can be used for configuring, instantiating, and managing SR policies. The model is generic and apply equally to the MPLS and SRv6 instantiations of SR policies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Specification of Requirements | 3 |
| 3. Building Blocks | 3 |
| 4. YANG Model | 4 |
| 4.1. Types and Definitions | 5 |
| 4.2. SR Policy | 6 |
| 4.2.1. Configuration | 6 |
| 4.2.2. State | 10 |
| 4.2.3. Notification | 12 |
| 5. Pending Items | 13 |
| 6. YANG Specification | 14 |
| 6.1. Types | 14 |
| 6.2. SR Policy | 22 |
| 7. Security Considerations | 44 |
| 8. IANA Considerations | 44 |
| 9. Acknowledgments | 45 |
| 10. References | 45 |
| 10.1. Normative References | 45 |
| 10.2. Informative References | 46 |
| Authors' Addresses | 46 |

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [RFC8402], allows a headend node to steer a packet flow along any topological path and/or service chain. The headend node is said to steer a flow into a Segment Routing Policy (SR Policy). An SR policy is a framework [I-D.ietf-spring-segment-routing-policy] that enables instantiation of an ordered list of segments on a node for implementing a policy.

This document introduces a YANG data model for SR policy framework for instantiating, configuring and managing SR policies along with its attributes. It is also expected that other companion models, such as BGP SR Policy [I-D.ietf-idr-segment-routing-te-policy], will be defined and/or augmented accordingly in their respective areas.

This model defines the following constructs for managing an SR policy:

- o Configuration
- o Operational State
- o Notifications
- o Executables (Actions)

This document expects and requires the reader to be well familiar with the concepts and constructs of an SR policy [I-D.ietf-spring-segment-routing-policy] as well as the YANG modeling language and its presentation [RFC6020].

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Building Blocks

Before looking into the YANG model for SR policy, it is important to recall and highlight the major building blocks and constructs that constitute and contribute to an SR policy, as described in [I-D.ietf-spring-segment-routing-policy].

- o **policy:** specifies constructs to allow a headend node to setup SR path(s) as an ordered list of segments for a given color and endpoint. The endpoint and the color are used to automate the steering of service or transport routes on an SR Policy. For a given headend, the key for an SR policy is (color, endpoint) where endpoint is an IP address that could be also NULL.
- o **candidate-path:** is the unit for signalling of an SR Policy to a headend via protocols (such as PCEP, BGP, CLI etc.). A candidate path is either dynamic or explicit type, where an explicit candidate path is associated with one or more segment-lists and

dynamic candidate path expresses optimization objectives and set of constraints. An SR Policy is associated with one or more candidate paths and the preference of the candidate path is used to select the best candidate path for an SR Policy. A candidate path is valid if it is usable (e.g. when its constituents SIDs are reachable). An "active" candidate path is the selected path (for forwarding) that is valid and determined to be the best path of the SR Policy. A candidate path is keyed by (protocol-origin, originator, discriminator).

- o segment-list: specifies ordered list of segments to traverse, where a segment can be specified in various forms (refer section 4 of [I-D.ietf-spring-segment-routing-policy]). The list is sorted by the index of the segment. A segment-list is used and referred by an explicit type of candidate-path. A segment-list is keyed by its name.
- o binding-sid: An SR policy is associated with a BSID to provide benefits of scaling, network opacity and service independence.

4. YANG Model

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

For SR policy YANG specification, this document defines following new YANG modules:

| Module Name | Purpose |
|----------------------|---|
| ietf-sr-policy-types | defines common and basic types related to an SR policy and related constructs |
| ietf-sr-policy | defines the model for SR policy instantiation, configuration, and management |

4.1. Types and Definitions

SR policy common types and definitions are defined in the new module "ietf-sr-policy-types". The main types defined in this module include:

- o dataplane-type: A union to specify MPLS or IPv6 as the dataplane type for SR.
- o sid-value-type: A Union to specify SID value for SR-MPLS or SRv6 type.
- o binding-sid-alloc-mode: Enum to define explicit or dynamic alloc mode types for a BSID.
- o protocol-origin-type: Enum to specify protocol origin (e.g. PCEP) for an SR policy.
- o explicit-binding-sid-rule-type: Enum to specify BSID alloc enforcement/rule when doing explicit alloc request.
- o binding-sid-oper-state: An Enum representing various operational states for a BSID.
- o policy-admin-state: An Enum for admin state of an SR policy.
- o policy-oper-state: An Enum for operational state of an SR policy.
- o segment-type: An Enum that defines various types for a "segment" of a Segment list.
- o candidate-path-non-selection-reason: The base identity along with its children to specify reason for not selecting a candidate path as the best/active path.
- o path-disjointness: The base identity for disjoint path computation. The disjointness types include link, node, srlg, srlg-node etc.
- o policy-down-reason: The base identity along with its children to specify reason for a policy becoming (or remaining) operationally down.
- o binding-sid-unavailable-reason: The base identity along with its children to specify reason for a BSID's unavailability.

The associated YANG specification for this module is captured in Section 6.1.

4.2. SR Policy

The base SR policy model is captured in `ietf-sr-policy` module. This base module augments `"/rt:routing"` and specifies the configuration, operational state, executables/rpcs, and notification events required to manage SR policies.

The associated YANG specification for this module is captured in Section 6.2.

4.2.1. Configuration

In terms of configuration hierarchy, SR policy configuration tree has following two main areas:

- o `attributes`: container that defines common constructs that could be used across policies. Examples of such a construct include `segment-lists`, `affinity-map` etc. In future revision of this document, it is expected that this container will have more constructs defined.
- o `policies`: container that defines list of policies with their attributes such as `BSID`, `candidate-paths` etc.

Following diagram depicts high level yang organization and hierarchy for an SR policy specification:

```

segment-routing
  traffic-engineering
    + attributes
    |
    | + affinity-map
    | | ....
    |
    | + segment-lists
    | | segment-list* [name]
    | | | segments
    | | | | segment* [index]
    | | | | ...
    | | + explicit-binding-sid-rules
    | | | ...
    |
    + policies
      policy* [color endpoint]
      + ...
      |
      + binding-sid
      | ...
      |
      + candidate-paths
      | candidate-path* [protocol-origin originator discriminator
      |
      | + ...
      | |
      | + type
      | | + explicit
      | | | segment-lists
      | | | | segment-list* [ref]
      | | | | ...
      | | + dynamic
      | | | constraints
      | | | ...
      |
      + ...

```

Figure 1: SR Policy - Hierarchy

Using the building blocks described in Section 3, following is the complete graphical representation of the data model for SR policy configuration:

```

module: ietf-sr-policy
  augment /rt:routing:
    +--rw segment-routing
    |   +--rw traffic-engineering
    |   |   +--rw attributes
    |   |   |   +--rw affinity-map

```

| | | | | |
|------|--|--|--|---|
| | | | | <pre> +--rw affinity* [name] +--rw name string +--rw bit-position? uint16 +--rw segment-lists +--rw segment-list* [name] +--rw name string +--rw segments +--rw segment* [index] +--rw index uint32 +--rw type? sr-policy-types:segment-type +--rw segment-types +--rw segment-type-1 +--rw sid-value? rt-types:mpls-label +--rw segment-type-2 +--rw sid-value? srv6-types:srv6-sid +--rw segment-type-3 +--rw ipv4-address? inet:ipv4-address +--rw algorithm? uint8 +--rw segment-type-4 +--rw ipv6-address? inet:ipv6-address +--rw algorithm? uint8 +--rw segment-type-5 +--rw ipv4-address? inet:ipv4-address +--rw interface-identifier? uint32 +--rw segment-type-6 +--rw local-ipv4-address? inet:ipv4-address +--rw remote-ipv4-address? inet:ipv4-address +--rw segment-type-7 +--rw local-ipv6-address? inet:ipv6-add </pre> |
| ress | | | | <pre> +--rw local-interface-identifier? uint32 +--rw remote-ipv6-address? inet:ipv6-add </pre> |
| ress | | | | <pre> +--rw remote-interface-identifier? uint32 +--rw segment-type-8 +--rw local-ipv6-address? inet:ipv6-address +--rw remote-ipv6-address? inet:ipv6-address +--rw segment-type-9 +--rw ipv6-address? inet:ipv6-address +--rw algorithm? uint8 +--rw segment-type-10 +--rw local-ipv6-address? inet:ipv6-add </pre> |
| ress | | | | <pre> +--rw local-interface-identifier? uint32 +--rw remote-ipv6-address? inet:ipv6-add </pre> |
| ress | | | | <pre> +--rw remote-interface-identifier? uint32 +--rw segment-type-11 +--rw local-ipv6-address? inet:ipv6-address +--rw remote-ipv6-address? inet:ipv6-address +--rw validate? boolean +--rw explicit-binding-sid-rules* [index] </pre> |

```

|      +---rw index      uint32
|      +---rw rule?      sr-policy-types:explicit-binding-sid-rule-type
+---rw policies
|   +---rw policy* [color endpoint]
|   |   +---rw color      uint32
|   |   +---rw endpoint   inet:ip-address
|   |   +---rw name?      string
|   |   +---rw description? string
|   |   +---rw admin-state? sr-policy-types:policy-admin-state
|   |   +---rw priority?   uint8
|   |   +---rw binding-sid
|   |   |   +---rw dataplane? sr-policy-types:dataplane-type
|   |   |   +---rw value?     sr-policy-types:sid-value-type
|   +---rw candidate-paths
|   |   +---rw candidate-path* [protocol-origin originator discriminato
r]
|   |   |   +---rw protocol-origin      sr-policy-types:protocol-or
igin-type
|   |   |   +---rw originator           string
|   |   |   +---rw discriminator        uint32
|   |   |   +---rw preference           uint32
|   |   |   +---rw name?               string
|   |   |   +---rw description?        string
|   |   |   +---rw binding-sid {capability-candidate-path-binding-sid}?
|   |   |   |   +---rw dataplane?      sr-policy-types:dataplane-type
|   |   |   |   +---rw value?         sr-policy-types:sid-value-type
|   |   |   +---rw (type)?
|   |   |   |   +---:(explicit)
|   |   |   |   |   +---rw segment-lists
|   |   |   |   |   |   +---rw segment-list* [name-ref]
|   |   |   |   |   |   |   +---rw name-ref    -> /rt:routing/sr-policy:segm
ent-routing/traffic-engineering/attributes/segment-lists/segment-list/name
|   |   |   |   |   |   +---rw weight?      uint32
|   |   |   |   +---:(dynamic)
|   |   |   |   |   +---rw sid-dataplane-type? sr-policy-types:datap
lane-type
|   |   +---rw constraints
|   |   |   +---rw affinities
|   |   |   |   +---rw exclude-any*   string
|   |   |   |   +---rw include-any*   string
|   |   |   |   +---rw include-all*   string
|   |   |   +---rw bounds
|   |   |   |   +---rw igp-metric-bound?   uint32
|   |   |   |   +---rw te-metric-bound?    uint32
|   |   |   |   +---rw latency-metric-bound? uint32
|   |   |   |   +---rw segment-bound?      uint32
|   |   |   +---rw segment-rules
|   |   |   |   +---rw sid-algorithm?   uint8
|   |   +---rw disjoint-path
|   |   |   +---rw group-id?           uint32
|   |   |   +---rw disjointness-type?   identityref
|   |   |   +---rw subgroup-id?        uint32

```


Figure 2: SR Policy - Config Tree

Please take note of the following important points in the above configuration model:

- o This model supports both MPLS and SRv6 dataplane for SR -- i.e. items like segments and BSID can be defined as MPLS label or SRv6 SIDs.
- o Specification of a segment supports all the types defined in SR policy base specification document
- o The above model supports explicit BSID specification on SR policy level as the main mode of specification. The model also allows explicit BSID per candidate-path as an if-feature capability that is optional for implementations
- o The above model will be extended in future revisions of this document to enhance constraints specification for dynamic type of candidate-path, as well as add traffic-steering controls.

4.2.2. State

As per NMDA model, the state related to configuration items specified in earlier Section 4.2.1 can be retrieved from the same tree. This section defines the other operational state items related to SR policy.

In addition to configured state, the operational state corresponding to the SR policy includes:

- o policy operational state
- o policy up/down timestamps
- o policy BSID info such as alloc mode, actual value in-use, operational state, and forwarding stats
- o Per candidate-path info such as:
 - * Whether candidate-path is the best candidate-path
 - * In case of non-best, the reason for such non-selection
 - * Type of candidate-path - explicit or dynamic
 - * Per segment-list information - such as validity of the segment-list, as well as forwarding state for a valid segment-list.

The forwarding state is represented in terms of per forwarding path info that includes nexthop address, outgoing interface, protection information, and encapsulation (label stack or SRv6 SID stack) etc.

Following is a simplified graphical representation of the data model for the SR policy (derived) operational state:

```

module: ietf-sr-policy
augment /rt:routing:
  +--rw segment-routing
    +--rw traffic-engineering
      +--rw policies
        +--rw policy* [color endpoint]
          +--rw color                uint32
          +--rw endpoint              inet:ip-address
          +--ro oper-state?           sr-policy-types:policy-oper-state
          +--ro transition-count?     uint32
          +--ro up-time?              yang:date-and-time
          +--ro down-time?            yang:date-and-time
          +--rw binding-sid
            +--ro alloc-mode?         sr-policy-types:binding-sid-alloc-mode
            +--ro allocated-sid?      sr-policy-types:sid-value-type
            +--ro oper-state?         sr-policy-types:binding-sid-oper-state
          +--ro counters
            +--ro pkts?               yang:counter64
            +--ro octets?             yang:counter64
          +--rw candidate-paths
            +--rw candidate-path* [protocol-origin originator discriminator]
              +--rw protocol-origin   sr-policy-types:protocol-or
              +--rw originator         string
              +--rw discriminator      uint32
              +--rw name                string
              +--ro is-best-candidate-path? boolean
              +--ro non-selection-reason? identityref
              +--ro is-valid?          boolean
              +--ro forwarding-paths
                +--ro forwarding-path* [path-id]
                  +--ro path-id        uint8
                  +--ro next-hop-address? inet:ip-address
                  +--ro next-hop-table-id? uint32
                  +--ro interface?      if:interface-ref
                  +--ro sid-list
                    +--ro (dataplanetype)?
                      +--:(mpls)
                        +--ro labels* [label]
                          +--ro label    rt-types:mpls-label

```

```

|      +--:(srv6)
|          +--ro sids* [sid]
|              +--ro sid      srv6-types:srv6-sid
+--ro is-protected?          boolean
+--ro is-pure-backup?        boolean
+--ro backup-path-id?        uint8
+--ro weight?                uint32

```

Figure 3: SR Policy - State Tree

4.2.3. Notification

This model defines a list of notifications to inform an operator of important events detected regarding an SR policy. These events include events related to:

- o policy status: policy operational state changes
- o Candidate-path active status and changes
- o Explicit Binding SID collision/unavailability events

Following is a simplified graphical representation of the data model for SR policy notifications:

```
module: ietf-sr-policy
```

```

  notifications:
    +---n sr-policy-oper-state-change-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-routing/traff
    |   affic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-routing/traff
    |   affic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-routing/traff
    |   affic-engineering/policies/policy/endpoint
    |   +---ro policy-new-oper-state?    sr-policy-types:policy-oper-state
    |   +---ro policy-down-reason?       identityref
    +---n sr-policy-candidate-path-change-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-routing/traff
    |   fic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-routing/traff
    |   fic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-routing/traff
    |   fic-engineering/policies/policy/endpoint
    |   +---ro existing-preference?      uint32
    |   +---ro new-preference?           uint32
    +---n sr-policy-binding-sid-unavailable-event
    |   +---ro policy-name-ref?          -> /rt:routing/sr-policy:segment-rout
    |   ing/traffic-engineering/policies/policy/name
    |   +---ro policy-color-ref?         -> /rt:routing/sr-policy:segment-rout
    |   ing/traffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?      -> /rt:routing/sr-policy:segment-rout
    |   ing/traffic-engineering/policies/policy/endpoint
    |   +---ro policy-binding-sid-value-ref? -> /rt:routing/sr-policy:segment-rout
    |   ing/traffic-engineering/policies/policy/binding-sid/value
    |   +---ro reason?                   identityref
    +---n sr-policy-candidate-path-binding-sid-mismatch-event
    |   +---ro policy-color-ref?          -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/color
    |   +---ro policy-endpoint-ref?       -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/endpoint
    |   +---ro existing-candidate-path-protocol-origin-ref? -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/can
    |   didate-path/protocol-origin
    |   +---ro existing-candidate-path-preference-ref?       -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/can
    |   didate-path/preference
    |   +---ro existing-candidate-path-binding-sid-dataplane-ref? -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/can
    |   didate-path/binding-sid/dataplane
    |   +---ro existing-candidate-path-binding-sid-value-ref? -> /rt:routing/
    |   sr-policy:segment-routing/traffic-engineering/policies/policy/candidate-paths/can
    |   didate-path/binding-sid/value
    |   +---ro conflicting-candidate-path-protocol-origin?   uint8
    |   +---ro conflicting-candidate-path-preference?         uint32
    |   +---ro conflicting-candidate-path-binding-sid-dataplane? sr-policy-types
    |   :dataplane-type
    |   +---ro conflicting-candidate-path-binding-sid-value?   sr-policy-types
    |   :sid-value-type

```

Figure 4: SR Policy - Notification Tree

5. Pending Items

Following are the items that will be addressed in future revisions of this document:

- o Configuration and Specification of:
 - * Traffic steering over SR policy
 - * ODN templates
 - * Spray policy

- o Executables (RPC actions)

6. YANG Specification

Following are actual YANG definition for the modules defined earlier in the document.

6.1. Types

<CODE BEGINS> file "ietf-sr-policy-types@2019-11-04.yang" -->

```
module ietf-sr-policy-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy-types";

  prefix "sr-policy-types";

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  import ietf-srv6-types {
    prefix "srv6-types";
  }

  organization "IETF SPRING Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/spring/>
    WG List:    <mailto:spring@ietf.org>
    Editor:     Kamran Raza
                <mailto:skraza@cisco.com>
    Editor:     Zhuang Shunwan
                <mailto:zhuangshunwa@huawei.com>
    Editor:     Daniel Voyer
                <mailto:daniel.voyer@bell.ca>
    Editor:     Muhammad Durrani
                <mailto:mdurrani@equinix.com>
    Editor:     Satoru Matsushima
                <mailto:satoru.matsushima@g.softbank.co.jp>
    Editor:     Pavan Vishnu Beeram
                <mailto:vbeeram@juniper.net>
    ";
```

```
description
  "This YANG module defines the essential types for the management
  of SR policy module.
  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.
  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).";

revision "2019-11-04" {
  description
    "New editor added";
  reference
    "draft-raza-spring-sr-policy-yang";
}

revision "2019-07-08" {
  description
    "Dynamic TE candidate-path support";
  reference
    "draft-raza-spring-sr-policy-yang";
}

revision "2018-07-01" {
  description
    "Initial version";
  reference
    "draft-raza-spring-sr-policy-yang";
}

/* Identities */
identity candidate-path-not-selected-reason {
  description
    "Base identity for which reasons for not selecting
    candidate path are derived from";
}
identity candidate-path-not-selected-not-best {
  base candidate-path-not-selected-reason;
  description
    "Higher preference path exists";
}
identity candidate-path-not-selected-no-valid-segment-list {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has no valid segment list(s)";
}
```

```
}
identity candidate-path-not-selected-empty-segment-list {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has empty segment list(s)";
}
identity candidate-path-not-selected-invalid-binding-sid {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has invalid binding SID";
}

identity policy-down-reason {
  description
    "Base identity for the reasons why SR policy is operationally down";
}
identity policy-down-reason-admin-down {
  base policy-down-reason;
  description "Policy is administrately down";
}
identity policy-down-reason-no-source-address {
  base policy-down-reason;
  description "Policy has no source address";
}
identity policy-down-reason-no-endpoint {
  base policy-down-reason;
  description "Policy has no end-point";
}
identity policy-down-reason-no-candidate-path {
  base policy-down-reason;
  description "Policy has no candidate path";
}
identity policy-down-reason-no-valid-candidate-path {
  base policy-down-reason;
  description "Policy has no valid candidate path";
}
identity policy-down-reason-candidate-path-invalid-segment-list {
  base policy-down-reason;
  description "Policy's candidate path has invalid segment list";
}
identity policy-down-reason-policy-unconfigured {
  base policy-down-reason;
  description "Policy is unconfigured";
}
identity policy-down-reason-policy-color-endpoint-updated {
  base policy-down-reason;
  description "Policy's color and end-point are updated";
}
```



```
identity policy-down-reason-local-label-setup-failed {
  base policy-down-reason;
  description "Policy's local label setup (allocation/rewrite) failed";
}
identity policy-down-reason-forwarding-rewrite-failed {
  base policy-down-reason;
  description "Policy's forwarding rewrite installation failed";
}
identity policy-down-reason-internal-error {
  base policy-down-reason;
  description "Infra related internal error";
}

identity binding-sid-unavailable-reason {
  description
    "Base identity for binding sid unavailable reason types";
}
identity binding-sid-allocation-error {
  base binding-sid-unavailable-reason;
  description "SID allocator returned an error";
}
identity binding-sid-already-exists {
  base binding-sid-unavailable-reason;
  description "Binding sid already exists/allocated";
}
identity binding-sid-internal-error {
  base binding-sid-unavailable-reason;
  description "Internal error with binding sid allocation";
}
identity binding-sid-color-endpoint-conflict {
  base binding-sid-unavailable-reason;
  description "Binding sid already allocated by another sr-policy with differen
t color/endpoint";
}
identity binding-sid-rewrite-error {
  base binding-sid-unavailable-reason;
  description "Binding sid forwarding rewrite error";
}
identity binding-sid-outside-srlb-range {
  base binding-sid-unavailable-reason;
  description "Binding sid outside SRLB range";
}

identity path-disjointness {
  description
    "Base identity for the type of path disjointness computation";
}
identity path-disjointness-link {
  base path-disjointness;
```

```
    description "The computed path is link-disjoint with the existing path";
  }
  identity path-disjointness-node {
    base path-disjointness;
    description "The computed path is node-disjoint with the existing path";
  }
  identity path-disjointness-srlg {
    base path-disjointness;
    description "The computed path is srlg-disjoint with the existing path";
  }
  identity path-disjointness-srlg-node {
    base path-disjointness;
    description "The computed path is node and srlg disjoint with the existing pa
th";
  }

/* Typdefs */
typedef sid-value-type {
  type union {
    type rt-types:mpls-label;
    type srv6-types:srv6-sid;
  }
  description "The SID value type";
}

typedef binding-sid-oper-state {
  type enumeration {
    enum ALLOC-PENDING {
      value 1;
      description "SID allocation pending for Binding SID";
    }
    enum PROGRAMMED {
      value 3;
      description "Binding SID is programmed in forwarding";
    }
    enum CONFLICT {
      value 4;
      description "Binding SID is in-conflict state with
regards to SID allocation. This also means that SID
allocation is pending";
    }
  }
  description
    "Binding SID operational state type";
}

typedef policy-admin-state {
  type enumeration {
    enum UP {
```

```
        value 1;
        description "SR policy is administratively up";
    }
    enum DOWN {
        value 2;
        description "SR policy is administratively down";
    }
}
description "SR policy admin state";
}

typedef policy-oper-state {
    type enumeration {
        enum UP {
            value 1;
            description "SR policy is operationally up";
        }
        enum DOWN {
            value 2;
            description "SR policy is operationally down";
        }
    }
    description "SR policy oper state";
}

typedef segment-type {
    type enumeration {
        enum segment-type-1 {
            value 1;
            description "SR-MPLS Label";
        }
        enum segment-type-2 {
            value 2;
            description "SRv6 SID";
        }
        enum segment-type-3 {
            value 3;
            description "IPv4 Prefix with optional SR Algorithm";
        }
        enum segment-type-4 {
            value 4;
            description "IPv6 Global Prefix with optional SR Algorithm for SR-MPLS";
        }
        enum segment-type-5 {
            value 5;
            description "IPv4 Prefix with Local Interface ID";
        }
        enum segment-type-6 {
```

```
        value 6;
        description "IPv4 Addresses for link endpoints as Local, Remote pair";
    }
    enum segment-type-7 {
        value 7;
        description "IPv6 Prefix and Interface ID for link endpoints as Local,
            Remote pair for SR-MPLS";
    }
    enum segment-type-8 {
        value 8;
        description "IPv6 Addresses for link endpoints as Local, Remote pair for
            SR-MPLS";
    }
    enum segment-type-9 {
        value 9;
        description "IPv6 Global Prefix with optional SR Algorithm for SRv6";
    }
    enum segment-type-10 {
        value 10;
        description "IPv6 Prefix and Interface ID for link endpoints as Local,
            Remote pair for SRv6";
    }
    enum segment-type-11 {
        value 11;
        description "IPv6 Addresses for link endpoints as Local, Remote pair for
            SRv6";
    }
}
description "SR segment type";
}

typedef dataplane-type {
    type enumeration {
        enum mpls {
            value 1;
            description "Segment-routing MPLS";
        }
        enum srv6 {
            value 2;
            description "Segment-routing v6";
        }
    }
}
description "Dataplane type of the segments";
}

typedef binding-sid-alloc-mode {
    type enumeration {
        enum explicit {
```

```
        value 1;
        description "Explicitly specified BSID";
    }
    enum dynamic {
        value 2;
        description "Dynamically allocated BSID";
    }
}
description "binding SID allocation mode";
}

typedef protocol-origin-type {
    type enumeration {
        enum pcep {
            value 10;
            description "PCEP used as signalling mechanism for the candidate path";
        }
        enum bgp {
            value 20;
            description "BGP used as signalling mechanism for the candidate path";
        }
        enum local {
            value 30;
            description "CLI, Yang model via Netconf, gRPC, etc used for candidate path instantiation";
        }
    }

    description "Originating protocol type";
}

typedef explicit-binding-sid-rule-type {
    type enumeration {
        enum enforce-srlb {
            value 1;
            description
                "Explicit Binding SID is enforced with no
                 fallback if label does not fall in SRLB or
                 if no SRLB is configured";
        }
        enum fallback-dynamic {
            value 2;
            description
                "Explicit Binding SID falls back to dynamic in
                 case explicit label is not available.";
        }
    }
    description "Explicit binding sid rule types";
}
```

```
} // module
```

```
<CODE ENDS>
```

Figure 5: ietf-sr-policy-types.yang

6.2. SR Policy

```
<CODE BEGINS> file "ietf-sr-policy@2019-11-04.yang" -->
```

```
module ietf-sr-policy {  
    namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy";  
    prefix "sr-policy";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ietf-interfaces {  
        prefix if;  
    }  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-routing-types {  
        prefix "rt-types";  
    }  
  
    import ietf-yang-types {  
        prefix "yang";  
    }  
  
    import ietf-srv6-types {  
        prefix "srv6-types";  
    }  
  
    import ietf-sr-policy-types {  
        prefix "sr-policy-types";  
    }  
  
    organization "IETF SPRING Working Group";
```

contact

"WG Web: <<http://tools.ietf.org/wg/spring/>>
WG List: <<mailto:spring@ietf.org>>

Editor: Kamran Raza
<<mailto:skraza@cisco.com>>

Editor: Zhuang Shunwan
<<mailto:zhuangshunwa@huawei.com>>

Editor: Daniel Voyer
<<mailto:daniel.voyer@bell.ca>>

Editor: Muhammad Durrani
<<mailto:mdurrani@equinix.com>>

Editor: Satoru Matsushima
<<mailto:satoru.matsushima@g.softbank.co.jp>>

Editor: Pavan Vishnu Beeram
<<mailto:vbeeram@juniper.net>>
";

description

"This module contains a collection of YANG definitions
for SR policy module.

Copyright (c) 2019 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).";

```
revision "2019-11-04" {  
  description  
    "Changes in keys for policy and its candidate paths";  
  reference  
    "draft-raza-spring-sr-policy-yang";  
}
```

```
revision "2019-07-08" {  
  description  
    "Dynamic TE candidate-path support";
```

```
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  revision "2018-07-01" {
    description
      "Initial version";
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  grouping segment_config {
    description "Segment properties grouping";
    leaf index {
      type uint32;
      description "Segment index";
    }
    leaf type {
      type sr-policy-types:segment-type;
      description "Segment type";
    }
  }
  container segment-types {
    description "Types of segments";
    container segment-type-1 {
      description
        "Segment declared by MPLS label";
      leaf sid-value {
        type rt-types:mpls-label;
        description "MPLS label value";
      }
    }
    container segment-type-2 {
      description
        "Segment declared by SRv6 SID value";
      leaf sid-value {
        type srv6-types:srv6-sid;
        description "SRv6 SID value";
      }
    }
    container segment-type-3 {
      description
        "Segment declared by IPv4 Prefix with optional SR Algorithm";
      leaf ipv4-address {
        type inet:ipv4-address;
        description "Segment IPv4 address";
      }
      leaf algorithm {
        type uint8;
      }
    }
  }
```



```
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-4 {
    description
        "Segment declared by IPv6 Global Prefix with optional
        SR Algorithm for SR-MPLS";
    leaf ipv6-address {
        type inet:ipv6-address;
        description "Segment IPv6 address";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-5 {
    description
        "Segment declared by IPv4 Prefix with Local Interface ID";
    leaf ipv4-address {
        type inet:ipv4-address;
        description "Node IPv4 address";
    }
    leaf interface-identifier {
        type uint32;
        description "local interface identifier";
    }
}
container segment-type-6 {
    description
        "Segment declared by IPv4 Addresses for link endpoints
        as Local, Remote pair";
    leaf local-ipv4-address {
        type inet:ipv4-address;
        description "Segment local IPv4 adjacency address";
    }
    leaf remote-ipv4-address {
        type inet:ipv4-address;
        description "Segment remote IPv4 adjacency address";
    }
}
container segment-type-7 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for
        link endpoints as Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Local link IPv6 address";
    }
}
```

```
    }
    leaf local-interface-identifier {
        type uint32;
        description "Local interface identifier";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Remote link IPv6 address";
    }
    leaf remote-interface-identifier {
        type uint32;
        description "Remote interface identifier";
    }
}
container segment-type-8 {
    description
        "Segment declared by IPv6 Addresses for link endpoints as
        Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Segment remote IPv6 adjacency address";
    }
}
container segment-type-9 {
    description
        "Segment declared by IPv6 Global Prefix with optional
        SR Algorithm for SRv6";
    leaf ipv6-address {
        type inet:ipv6-address;
        description "Segment IPv6 prefix";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-10 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for
        link endpoints as Local, Remote pair for SRv6";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Local link IPv6 address";
    }
}
```

```
    leaf local-interface-identifier {
        type uint32;
        description "Local interface identifier";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Remote link IPv6 address";
    }
    leaf remote-interface-identifier {
        type uint32;
        description "Remote interface identifier";
    }
}
container segment-type-11 {
    description
        "Segment declared by IPv6 Addresses for link endpoints as
        Local, Remote pair for SRv6";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Segment remote IPv6 adjacency address";
    }
}
}
leaf validate {
    type boolean;
    default 'false';
    description "Indicates whether the segment should be validated. The default
        applies to all segments other than the first segment. For the
        first segment, validation is always done.";
}
}

grouping segment-properties {
    description
        "SR segment properties grouping";
    uses segment_config;
}

grouping attributes {
    description
        "Grouping containing attributes applicable to all SR policies";

    container attributes {
        description
```

```
    "Attributes applicable to SR policies";

    uses affinity-mapping;
    uses segment-lists;
    uses explicit-binding-sid-rules;
  }
}

grouping segment-lists {
  description
    "Segment lists grouping";
  container segment-lists {
    description "Segment-lists properties";

    list segment-list {
      key "name";
      description "Segment-list properties";
      leaf name {
        type string;
        description "Segment-list name";
      }
      container segments {
        description
          "Segments for given segment list";

        list segment {
          key "index";
          description "Configure Segment/hop at the index";
          uses segment-properties;
        }
      }
    }
  }
}

grouping binding-sid_config {
  description
    "Binding SID configuration properties grouping";
  leaf dataplane {
    type sr-policy-types:dataplane-type;
    description "Binding SID dataplane type";
  }
  leaf value {
    type sr-policy-types:sid-value-type;
    description "Binding SID value";
  }
}
```

```
grouping forwarding-counters {
  description
    "Grouping for counters";
  container counters {
    config false;
    description
      "Counters containing stats related to forwarding";

    leaf pkts {
      type yang:counter64;
      description "Number of packets forwarded";
    }
    leaf octets {
      type yang:counter64;
      units "byte";
      description "Number of bytes forwarded";
    }
  }
}

grouping binding-sid_state {
  description
    "Binding SID state properties grouping";
  leaf alloc-mode {
    type sr-policy-types:binding-sid-alloc-mode;
    config false;
    description "Binding SID type";
  }
  leaf allocated-sid {
    type sr-policy-types:sid-value-type;
    config false;
    description "Allocated SID value for the Binding SID";
  }
  leaf oper-state {
    type sr-policy-types:binding-sid-oper-state;
    config false;
    description
      "Binding SID operational state";
  }
}

grouping binding-sid-properties {
  description
    "Binding SID properties grouping";
  container binding-sid {
    description "Binding Segment ID";
    uses binding-sid_config;
    uses binding-sid_state;
  }
}
```

```
    }
  }

  grouping mpls-label-stack {
    description
      "Grouping for MPLS label stack";

    list labels {
      key "label";
      description
        "Stack containing MPLS labels";

      leaf label {
        type rt-types:mpls-label;
        description
          "MPLS label value";
      }
    }
  }

  grouping srv6-sid-stack {
    description
      "Grouping for SRv6 label stack";

    list sids {
      key "sid";
      description
        "Stack containing SRv6 SIDs";

      leaf sid {
        type srv6-types:srv6-sid;
        description
          "SRv6 sid value";
      }
    }
  }

  grouping path-forwarding_state {
    description "Policy Forwarding path information";
    leaf path-id {
      type uint8;
      description "Primary path id";
    }
    leaf next-hop-address {
      type inet:ip-address;
      description "Nexthop address";
    }
    leaf next-hop-table-id {
```

```
    type uint32;
    description "Table ID for nexthop address";
  }
  leaf interface {
    type if:interface-ref;
    description "Outgoing interface handle";
  }
  container sid-list {
    description
      "Outgoing sid stack";
    choice dataplanetype {
      description
        "Outgoing sids dataplane choice";
      case mpls {
        uses mpls-label-stack;
      }
      case srv6 {
        uses srv6-sid-stack;
      }
    }
  }
}
leaf is-protected {
  type boolean;
  description "Is this path protected ?";
}
leaf is-pure-backup {
  type boolean;
  description "Is this path a pure backup ?";
}
leaf backup-path-id {
  type uint8;
  description "Backup path id";
}
leaf weight {
  type uint32;
  description "Path's weight for W-ECMP balancing";
}
}

grouping cpath-cmn-properties {
  description
    "Common properties of the candidate path";

  leaf is-valid {
    type boolean;
    config false;
    description
      "True if the segment-list is valid, False otherwise";
  }
}
```

```

    }

    container forwarding-paths {
        config false;
        description
            "Forwarding state of paths";
        list forwarding-path {
            key "path-id";
            description "Forwarding path";
            uses path-forwarding_state;
        }
    }
}

grouping explicit-path-properties {
    description
        "Explicit path properties of the candidate path";
    container segment-lists {
        description
            "Path segment list(s) properties";
        list segment-list {
            key "name-ref";
            description "SR policy candidate path segment lists";

            leaf name-ref {
                type leafref {
                    path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-enginee
ring/sr-policy:attributes/sr-policy:segment-lists/sr-policy:segment-list/sr-polic
y:name";
                }
                description "Reference to segment-list name";
            }
            leaf weight {
                type uint32;
                description "Segment-list weighted loadshare";
            }
        }
    }
}

grouping affinity-mapping {
    description "Affinity-map grouping";

    container affinity-map {
        description
            "Mapping of affinity names to bit position";
        list affinity {
            key "name";
            unique "bit-position";
            leaf name {

```



```

        type string;
        description
            "Name of the affinity";
    }
    leaf bit-position {
        type uint16;
        description
            "The affinity entry in this list is mapped to the this bit-position i
n the
        affinity bitmap";
    }

    description "Affinity";
}
}
}
grouping dynamic-path-properties {
    description
        "Dynamic path properties of the candidate path";
    leaf sid-dataplane-type {
        type sr-policy-types:dataplane-type;
        description
            "The dataplane type for the sid";
    }

    container constraints {
        description "Constraints for the dynamic path computation";
        container affinities {
            description "Affinity constraints on the computed dynamic path";
            leaf-list exclude-any {
                type string;
                description
                    "The link is excluded if it has any of these affinities.";
            }
            leaf-list include-any {
                type string;
                description
                    "The link is accepted if it has any of these affinities";
            }
            leaf-list include-all {
                type string;
                description
                    "The link is accepted if it has all these affinities";
            }
        }

        container bounds {
            description "Upper-bound constraints on the computed dynamic path";
            leaf igp-metric-bound {

```

```
        type uint32;
        description
            "Path is invalid if its IGP metric exceeds this value";
    }
    leaf te-metric-bound {
        type uint32;
        description
            "Path is invalid if its TE metric exceeds this value";
    }
    leaf latency-metric-bound {
        type uint32;
        units "microsecond";
        description
            "Path is invalid if its latency exceeds this value";
    }
    leaf segment-bound {
        type uint32;
        description
            "Path is invalid if it has more segments than this value";
    }
}
container segment-rules {
    description "Constraints on the segments to be used in the path";
    leaf sid-algorithm {
        type uint8 {
            range "128..255";
        }
        description
            "The prefix-sid algorithm to be used in path calculation";
    }
}
container disjoint-path {
    description "Path disjointness constraints";
    leaf group-id {
        type uint32 { range "1..65535"; }
        description "";
    }
    leaf disjointness-type {
        type identityref { base sr-policy-types:path-disjointness; }
        description
            "Type of disjointness computation used to find the path";
    }
    leaf subgroup-id {
        type uint32 { range "1..65535"; }
        description "";
    }
}
}
```

```
}

grouping candidate-path_state {
  description
    "Candidate path state properties grouping";
  leaf is-best-candidate-path {
    type boolean;
    default 'false';
    config false;
    description
      "True if the candidate path is the best candidate path, False otherwise";
  }
  leaf non-selection-reason {
    type identityref {
      base sr-policy-types:candidate-path-not-selected-reason;
    }
    config false;
    description
      "Candidate path not selected reason";
  }
}

grouping policy-properties_config {
  description
    "SR policy configuration grouping";
  leaf name {
    type string {
      length "1..59";
    }
    description "SR policy name";
  }
  leaf color {
    type uint32 {
      range "1..4294967295";
    }
    description "Color associated with the policy";
  }
  leaf endpoint {
    type inet:ip-address;
    description "Policy end point IP address";
  }
  leaf description {
    type string;
    description "Description of the policy";
  }
  leaf admin-state {
    type sr-policy-types:policy-admin-state;
    default 'UP';
  }
}
```

```
        description
            "SR policy administrative state, true for
            enabled, false for disabled";
    }
}

grouping policy-properties_state {
    description
        "SR policy property grouping";
    leaf oper-state {
        type sr-policy-types:policy-oper-state;
        config false;
        description
            "SR policy operational state";
    }
    leaf transition-count {
        type uint32;
        config false;
        description "Indicates number of up/down transitions";
    }
    leaf up-time {
        type yang:date-and-time;
        config false;
        description "Policy up time in seconds";
    }
    leaf down-time {
        type yang:date-and-time;
        config false;
        description "Policy down time in seconds";
    }
}

grouping policy-properties {
    description
        "SR policy properties";
    uses policy-properties_state;
    uses binding-sid-properties;
    uses forwarding-counters;
}

grouping candidate-path-type {
    description "Candidate path type grouping";
    choice type {
        description
            "Type of candidate paths";
        case explicit {
            description "Candidate path with explicitly defined set/s of segment-list
s";
            uses explicit-path-properties;
        }
    }
}
```

```
    }
    case dynamic {
      description "Candidate path with dynamic computed segment-lists";
      uses dynamic-path-properties;
    }
  }
}

grouping candidate-paths {
  description "SR policy candidate path grouping";
  container candidate-paths {
    description "SR policy candidate path(s) ";

    list candidate-path {
      key "protocol-origin originator discriminator";
      unique "preference";

      description "SR policy Candidate path(s) list entry";

      leaf protocol-origin {
        type sr-policy-types:protocol-origin-type;
        description
          "Instantiation mechanism used to create the candidate path";
      }
      leaf originator {
        type string;
        description
          "Identifier (concatenation of ASN and node-address) of the node
           that signalled/instantiated the candidate path on headend";
      }
      leaf discriminator {
        type uint32;
        description "Candidate path distinguisher";
      }

      leaf preference {
        type uint32 {
          range "1..65535";
        }
        mandatory true;
        description "Candidate path preference";
      }
      leaf name {
        type string;
        description "Candidate path name";
      }
      leaf description {
        type string;
      }
    }
  }
}
```

```
        description "Candidate path description";
    }
    container binding-sid {
        if-feature capability-candidate-path-binding-sid;
        description
            "Binding segment ID";
        uses binding-sid_config;
    }

    uses candidate-path-type;
    uses candidate-path_state;
    uses cpath-cmn-properties;
}
}

grouping policies {
    description "SR policy grouping";
    container policies {
        description "SR Policy container";

        list policy {
            key "color endpoint";
            unique "name";

            description "SR Policy properties";
            leaf color {
                type uint32 {
                    range "1..4294967295";
                }
                description "Color associated with the policy";
            }
            leaf endpoint {
                type inet:ip-address;
                description "Policy end point IP address";
            }
            leaf name {
                type string {
                    length "1..59";
                }
                description "SR policy name";
            }
            leaf description {
                type string;
                description "Description of the policy";
            }
            leaf admin-state {
                type sr-policy-types:policy-admin-state;
            }
        }
    }
}
```

```
        default 'UP';
        description
            "SR policy administrative state, true for
            enabled, false for disabled";
    }
    leaf priority {
        type uint8;
        default 128;
        description "Priority considered when policy is recomputed due to topology changes";
    }

    uses policy-properties;

    uses candidate-paths;
}

}

grouping explicit-binding-sid-rules {
    description
        "Grouping for explicit binding sid rules";

    list explicit-binding-sid-rules {
        key "index";
        description
            "Explicit binding sid rules applicable for all policies";
        leaf index {
            type uint32;
            description "Explicit binding SID rules list index";
        }
        leaf rule {
            type sr-policy-types:explicit-binding-sid-rule-type;
            description "Explicit binding sid rule";
        }
    }
}

augment "/rt:routing" {
    description
        "This augments routing-instance configuration with segment-routing sr-policy.";
    container segment-routing {
        description "Main segment routing container";
        container traffic-engineering {
            description "Traffic-engineering container";

            uses attributes;

            uses policies;
        }
    }
}
```

```
    }
  }
}

/* Notifications */

notification sr-policy-oper-state-change-event {
  description
    "Notification event when the operational state of the SR policy changes";

  leaf policy-name-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:name";
    }
    description "Reference to sr-policy name";
  }

  leaf policy-color-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:color";
    }
    description "Reference to sr-policy color";
  }

  leaf policy-endpoint-ref {
    type leafref {
      path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
    }
    description "Reference to sr-policy endpoint";
  }

  leaf policy-new-oper-state {
    type sr-policy-types:policy-oper-state;
    description "New operational state of the SR policy";
  }

  leaf policy-down-reason {
    type identityref {
      base sr-policy-types:policy-down-reason;
    }
    description "Down reason if the SR policy's new operational state is down";
  }
}

notification sr-policy-candidate-path-change-event {
  description
    "Notification event when candidate path changes for SR policy";
}
```



```
    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-preference {
      type uint32;
      description "Existing candidate path preference";
    }

    leaf new-preference {
      type uint32;
      description "New candidate path preference";
    }
  }

  notification sr-policy-binding-sid-unavailable-event {
    description
      "Notification event when the binding sid of sr-policy is unavailable";

    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }
  }
}
```

```
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf policy-binding-sid-value-ref {
      if-feature capability-candidate-path-binding-sid;
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:binding-sid/sr-policy:value";
      }
      description "Reference to sr-policy binding-sid value";
    }

    leaf reason {
      type identityref {
        base sr-policy-types:binding-sid-unavailable-reason;
      }
      description
        "Reason why the binding sid is unavailable";
    }
  }

  notification sr-policy-candidate-path-binding-sid-mismatch-event {
    description
      "Notification event when binding sid of requested candidate path
      is different from the binding sid of the existing candidate path";

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-candidate-path-protocol-origin-ref {
      type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/
sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate
-path/sr-policy:protocol-origin";
      }
    }
  }
}
```



```
    }
    description "Reference to existing candidate path protocol origin";
}

leaf existing-candidate-path-preference-ref {
    type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:preference";
    }
    description "Reference to existing candidate path preference";
}

leaf existing-candidate-path-binding-sid-dataplane-ref {
    if-feature capability-candidate-path-binding-sid;
    type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:binding-sid/sr-policy:dataplane";
    }
    description "Reference to existing candidate path binding sid dataplane type";
}

leaf existing-candidate-path-binding-sid-value-ref {
    if-feature capability-candidate-path-binding-sid;
    type leafref {
        path "/rt:routing/sr-policy:segment-routing/sr-policy:traffic-engineering/sr-policy:policies/sr-policy:policy/sr-policy:candidate-paths/sr-policy:candidate-path/sr-policy:binding-sid/sr-policy:value";
    }
    description "Reference to existing candidate path binding sid value";
}

leaf conflicting-candidate-path-protocol-origin {
    type uint8;
    description "Conflicting candidate path protocol origin";
}

leaf conflicting-candidate-path-preference {
    type uint32;
    description "Conflicting candidate path preference";
}

leaf conflicting-candidate-path-binding-sid-dataplane {
    type sr-policy-types:dataplane-type;
    description "Conflicting candidate path binding sid dataplane type";
}

leaf conflicting-candidate-path-binding-sid-value {
    type sr-policy-types:sid-value-type;
    description "Conflicting candidate path binding sid value";
}
}
```



```

/* Features */

feature capability-candidate-path-binding-sid {
  description
    "This feature enables the capability of specifying binding-sid
    for a candidate path.";
}
} // module

<CODE ENDS>

```

Figure 6: ietf-sr-policy.yang

7. Security Considerations

The configuration, state, and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC8341].

8. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

| URI | Registrant | XML |
|--|------------|-----|
| urn:ietf:params:xml:ns:yang:ietf-sr-policy-types | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-sr-policy | The IESG | N/A |

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

| Name | Namespace | Prefix | Reference |
|----------------------|--|-----------------|---------------|
| ietf-sr-policy-types | urn:ietf:params:xml:ns:yang:ietf-sr-policy-types | sr-policy-types | This document |
| ietf-sr-policy | urn:ietf:params:xml:ns:yang:ietf-sr-policy | sr-policy | This document |

9. Acknowledgments

The authors of this document/YANG model would like to acknowledge the contributions/reviews by Johnson Thomas, Clarence Filsfils, Siva Sivabalan, Tarek Saad, Kris Michielsen, Dhanendra Jain, Ketan Talaulikar, Bhupendra Yadav, and Bruno Decraene.

10. References

10.1. Normative References

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-08 (work in progress), July 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.

10.2. Informative References

- [I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Rosen, E., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-09 (work in progress), May 2020.

Authors' Addresses

Kamran Raza (editor)
Cisco Systems
Email: skraza@cisco.com

Robert Sawaya
Cisco Systems
Email: rsawaya@cisco.com

Zhuang Shunwan
Huawei Technologies
Email: zhuangshunwa@huawei.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Muhammad Durrani
Equinix
Email: mdurrani@equinix.com

Satoru Matsushima
SoftBank
Email: satoru.matsushima@g.softbank.co.jp

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2020

K. Raza
S. Agarwal
Cisco Systems

X. Liu
Volta Networks

Z. Hu
Huawei Technologies

I. Hussain
Infinera Corporation

H. Shah
Ciena Corporation

D. Voyer
Bell Canada

H. Elmalky

S. Matsushima
K. Horiba
SoftBank

A. AbdelSalam
J. Rajamanickam
Cisco Systems

October 30, 2019

YANG Data Model for SRv6 Base and Static
draft-raza-spring-srv6-yang-05

Abstract

This document describes a YANG data model for Segment Routing IPv6 (SRv6) base. The model serves as a base framework for configuring and managing an SRv6 subsystem and expected to be augmented by other SRv6 technology models accordingly. Additionally, this document also specifies the model for the SRv6 Static application.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 2, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Specification of Requirements | 3 |
| 3. YANG Model | 4 |
| 3.1. Overview | 4 |
| 3.2. SRv6 Types | 4 |
| 3.3. SRv6 Base | 5 |
| 3.3.1. Configuration | 5 |
| 3.3.2. State | 6 |
| 3.3.3. Notification | 8 |
| 3.4. SRv6 Static | 9 |
| 3.4.1. Configuration | 9 |
| 3.4.2. State | 15 |
| 3.4.3. Notification | 15 |
| 4. Pending Items | 15 |
| 5. YANG Specification | 16 |
| 5.1. SRv6 Types | 16 |
| 5.2. SRv6 Base | 34 |
| 5.3. SRv6 Static | 49 |
| 6. Security Considerations | 73 |
| 7. IANA Considerations | 74 |
| 8. Acknowledgments | 75 |
| 9. References | 75 |
| 9.1. Normative References | 75 |
| 9.2. Informative References | 77 |

| | |
|------------------------------|----|
| Authors' Addresses | 77 |
|------------------------------|----|

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [RFC8402], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain. When applied to ipv6 data-plane (i.e. SRv6), SR requires a type of routing header (SRH) in an IPv6 packet that is used to encode an ordered list of IPv6 addresses (SIDs). The active segment is indicated by the Destination Address of the packet, and the next segment is indicated by a pointer in the SRH [I-D.ietf-6man-segment-routing-header]. The various functions and behaviors corresponding to network programming using SRv6 are specified in [I-D.ietf-spring-srv6-network-programming].

This document introduces a YANG data model for base SRv6 that would serve as a base framework for configuring and managing an SRv6 subsystem. As needed, other SRv6 technology models (e.g. ISIS, OSPFv3, BGP, EVPN, Service Chaining) may augment this model. Furthermore, to illustrate basic behaviors as captured in [I-D.ietf-spring-srv6-network-programming], this document also specifies a YANG model for the SRv6-Static application.

The model currently defines the following constructs that are used for managing SRv6:

- o Configuration
- o Operational State
- o Notifications

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. YANG Model

3.1. Overview

This document defines following three new YANG modules:

- o `ietf-srv6-types`: defines common and basic types related to SRv6
- o `ietf-srv6-base`: specifies management model for SRv6 base constructs (locator, SIDs, etc.)
- o `ietf-srv6-static`: specifies management model for SRv6-static application

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

In this document, when a simplified graphical representation of YANG model is presented in a tree diagram, the meaning of the symbols in these tree diagrams is defined in [RFC8340].

3.2. SRv6 Types

SRv6 common types and definitions are defined in the new module "ietf-srv6-types". The main types defined in this module include:

- o `srv6-sid`: SRv6 SID
- o `srv6-func-value`: Typedef for FUNC value in an SRv6 SID
- o `srv6-func-value-reserved-type`: Enum (list) of "reserved" FUNC opcode
- o `srv6-endpoint-type`: SRv6 Endpoint behaviors [I-D.ietf-spring-srv6-network-programming] identity type
- o `srv6-transit-type`: SRv6 Transit behavior types [I-D.ietf-spring-srv6-network-programming] identity type
- o `srv6-security-type`: SRv6 Security rule type [I-D.ietf-spring-srv6-network-programming] identity type

- o `srv6-counter-type`: SRv6 Counter type
[I-D.ietf-spring-srv6-network-programming] identity type

The corresponding YANG specification for this module is captured in Section 5.1.

3.3. SRv6 Base

The base SRv6 model is specified in `ietf-srv6-base` module. This module augments `"/rt:routing"` and specifies the configuration, operational state, and notification events that are required to manage the base SRv6.

The corresponding YANG specification for this module is captured in Section 5.2.

3.3.1. Configuration

The module defines some fundamental items required to configure an SRv6 network:

- o **SRv6 Enablement**: Enable Segment-Routing SRv6 feature
- o **Encapsulation Parameters**: Provide encapsulation related parameters (such as `source-address`, `hop-limit`, and `traffic-class`) to be used when performing `T.Encap*` operation.
- o **Locator(s) Specification**: SRv6 locator is a fundamental construct for an SRv6 network. This is the construct from which SID (function values) are allocated that on the local box, and advertised to and used by remote nodes for reachability. A locator is identified by a name and has associated prefix. It is possible to have more than one locator per node. In case of more than one locator, there is one and only one locator designated as the default locator.

Following is a simplified graphical tree representation of the data model for SRv6 base configuration

```

module: ietf-srv6-base
augment /rt:routing:
  +--rw srv6
    +--rw enable?                boolean
    +--rw encapsulation
      +--rw source-address?      inet:ipv6-address
      +--rw hop-limit
        +--rw value?            uint8
        +--rw propagate?        boolean
      +--rw traffic-class
        +--rw value?            uint8
        +--rw propagate?        boolean
    +--rw locators
      +--rw locator* [name]
        +--rw name                string
        +--rw enable?            boolean
        +--rw prefix
          +--rw address          inet:ipv6-address
          +--rw length           srv6-types:srv6-locator-len
        +--rw algorithm?         uint32

```

Figure 1: SRv6 Base - Config Tree

3.3.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.3.1 can be retrieved from the same tree. This section defines other operational state items related to SRv6 base.

The operational state corresponding to the SRv6 base includes:

- o node capabilities: provides information on the node (hardware) capabilities and support regarding various SRv6 aspects and features including end behaviors, transit behaviors, security rules, counter/stats support, and other SRv6 parameters that need to be signaled in an SRv6 network by the protocols.
- o locator: provides information related to a locator. The information includes locator operational state, and state of address conflict with any ipv6 address configured on local interfaces etc.
- o local-sid: provides information related to local-SIDs allocated and/or installed on the node. This includes two types of information:

1. aggregate across all local-SIDs such as aggregate counters
2. per local-SID information such as allocation type (dynamic or explicit), SID owner protocol(s)/client(s), forwarding [paths] information, and stats/counters.

Following is a simplified graphical tree representation of the data model for the SRv6 operational state (for read-only items):

```

module: ietf-srv6-base
augment /rt:routing:
  +--rw srv6
    +--rw locators
      +--rw locator* [name]
        +--rw name string
        +--ro operational-status? srv6-types:srv6-status-type
        +--ro is-in-address-conflict? boolean
      +--ro node-capabilities
        +--ro end-behavior* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro transit-behavior* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro signaled-parameters
          +--ro max-sl? uint8
          +--ro max-end-pop-srh? uint8
          +--ro max-t_insert? uint8
          +--ro max-t_encap? uint8
          +--ro max-end_d? uint8
        +--ro security-rule* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro counters* [type]
          +--ro type identityref
          +--ro supported boolean
      +--ro local-sids
        +--ro counters
          +--ro cnt-3
            +--ro in-pkts? yang:counter64
            +--ro in-octets? yang:counter64
          +--ro local-sid* [sid]
            +--ro sid srv6-types:srv6-sid
            +--ro locator? -> /rt:routing/srv6:srv6/locators/locato
r/name
          +--ro is-reserved? boolean
          +--ro end-behavior-type? identityref
          +--ro alloc-type? srv6-types:sid-alloc-type

```



```

+--ro owner* [type instance]
|   +--ro type          identityref
|   +--ro instance      string
|   +--ro is-winner?    boolean
+--ro forwarding
|   +--ro is-installed?  boolean
|   +--ro next-hop-type?  srv6-types:srv6-nexthop-type
|   +--ro paths
|       +--ro path* [path-index]
|           +--ro path-index  uint8
|           +--ro l2
|               | +--ro interface?  if:interface-ref
|           +--ro l3
|               | +--ro interface?          if:interface-ref
|               | +--ro next-hop?          inet:ip-address
|               | +--ro weight?            uint32
|               | +--ro role?              enumeration
|               | +--ro backup-path-index?  uint8
|           +--ro (encap-type)?
|               +--:(srv6)
|                   +--ro out-sid* [sid]
|                       +--ro sid      srv6-types:srv6-sid
|               +--:(mpls)
|                   +--ro out-label* [label]
|                       +--ro label    rt-types:mpls-label
+--ro counters
    +--ro cnt-1
        +--ro in-pkts?      yang:counter64
        +--ro in-octets?    yang:counter64

```

Figure 2: SRv6 Base - State Tree

3.3.3. Notification

This model defines a list of notifications to inform an operator of important events detected during the SRv6 operation. These events include events related to:

- o locator operational state changes
- o local-SID collision event

Following is a simplified graphical tree representation of the data model for SRv6 notifications:

```

module: ietf-srv6-base

  notifications:
    +---n srv6-locator-status-event
    |   +---ro operational-status?  srv6-types:srv6-status-type
    |   +---ro locator?             -> /rt:routing/srv6:srv6/locators/locator/nam
e
    +---n srv6-sid-collision-event
    |   +---ro sid?                 srv6-types:srv6-sid
    |   +---ro existing
    |   |   +---ro end-behavior-type?  identityref
    |   +---ro requested
    |       +---ro end-behavior-type?  identityref

```

Figure 3: SRv6 Base - Notification Tree

3.4. SRv6 Static

SRv6-Static application allows a user to specify SRv6 local SIDs and program them in the forwarding plane. The SRv6-Static model is captured in the ietf-srv6-static module.

The associated YANG specification for this module is captured in Section 5.3.

3.4.1. Configuration

The SRv6-Static configuration augments the SRv6-base locator tree `/rt:routing/srv6:srv6/srv6:locators/srv6:locator`

Following are salient features of the SRv6-Static config model:

- o Allows static (explicit) configuration for local-SIDs under a given locator
- o Given that entry is scoped under a locator, the key for each entry is "function" value.
- o A user must also specify end-behavior type (End* function) associated with the entry
- o A user must also specify behavior-specific data with each entry. For example, for any end behavior requiring a table lookup, a lookup-table need be provided. Similarly, for any end behavior with forwarding next-hops need to specify next-hop information. The example of former include End, End.T, End.DT4, End.DT6, and End.DT46, whereas example of later include End.X, End.DX4, End.DX6, End.B6, End.BM etc.

- o Each local-SID entry has zero or more forwarding paths specified.
- o A forwarding path has next-hop type that depends on the end behavior, and could be either ipv6, or ipv4, or mpls, or l2 type. For example, End.X, End.DX4, End.DX6, End.B6, End.BM, and End.DX2 will have ipv6, ipv4, ipv6, ipv6, mpls, and l2 next-hop types respectively
- o For each forwarding next-hop type, the appropriate path attributes are to be specified as well. For L2 type, the only other information required is the L2 interface name. Whereas for L3 (ipv6, ipv4, mpls) types, the information includes L3 interface name, next-hop IP address, weight, and protection information.
- o Depending on the end behavior type, a forwarding path may have either MPLS or SRv6 encapsulation -- i.e., Stack of out-labels or Stack of SRv6 out-SIDs. The example of former is End.BM and example of later include the rest (End.X, End.DX4/DX6, End.B6 etc.).

Following is a simplified graphical tree representation of the data model for SRv6 Static configuration

```

module: ietf-srv6-static
  augment /rt:routing/srv6:srv6/srv6:locators/srv6:locator:
    +--rw static
      +--rw local-sids
        +--rw sid* [function]
          +--rw function          srv6-types:srv6-func-value
          +--rw end-behavior-type identityref
          +--rw end
          +--rw end_psp
          +--rw end_usp
          +--rw end_psp_usp
          +--rw end_usd
          +--rw end_psp_usd
          +--rw end_usp_usd
          +--rw end_psp_usp_usd
          +--rw end-t
          | +--rw lookup-table-ipv6  srv6-types:table-id
          +--rw end-t_psp
          | +--rw lookup-table-ipv6  srv6-types:table-id
          +--rw end-t_usp
          | +--rw lookup-table-ipv6  srv6-types:table-id
          +--rw end-t_psp_usp
          | +--rw lookup-table-ipv6  srv6-types:table-id
          +--rw end-t_usd

```

```

|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-t_psp_usd
|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-t_usp_usd
|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-t_psp_usp_usd
|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-x
|   +---rw protected?      boolean
+---rw paths
|   +---rw path* [path-index]
|   |   +---rw path-index      uint8
|   |   +---rw interface?      if:interface-ref
|   |   +---rw next-hop?      inet:ipv6-address
|   |   +---rw table?          srv6-types:table-id
|   |   +---rw weight?         uint32
|   |   +---rw role?           enumeration
|   |   +---rw backup-path-index?  uint8
|   |   +---rw sid-list
|   |   |   +---rw out-sid* [sid]
|   |   |   |   +---rw sid      srv6-types:srv6-sid
+---rw end-x_psp
|   +---rw protected?      boolean
+---rw paths
|   +---rw path* [path-index]
|   |   +---rw path-index      uint8
|   |   +---rw interface?      if:interface-ref
|   |   +---rw next-hop?      inet:ipv6-address
|   |   +---rw table?          srv6-types:table-id
|   |   +---rw weight?         uint32
|   |   +---rw role?           enumeration
|   |   +---rw backup-path-index?  uint8
|   |   +---rw sid-list
|   |   |   +---rw out-sid* [sid]
|   |   |   |   +---rw sid      srv6-types:srv6-sid
+---rw end-x_usp
|   +---rw protected?      boolean
+---rw paths
|   +---rw path* [path-index]
|   |   +---rw path-index      uint8
|   |   +---rw interface?      if:interface-ref
|   |   +---rw next-hop?      inet:ipv6-address
|   |   +---rw table?          srv6-types:table-id
|   |   +---rw weight?         uint32
|   |   +---rw role?           enumeration
|   |   +---rw backup-path-index?  uint8
|   |   +---rw sid-list
|   |   |   +---rw out-sid* [sid]

```

```

|               +--rw sid      srv6-types:srv6-sid
+--rw end-x_psp_osp
|   +--rw protected?    boolean
|   +--rw paths
|       +--rw path* [path-index]
|           +--rw path-index      uint8
|           +--rw interface?      if:interface-ref
|           +--rw next-hop?       inet:ipv6-address
|           +--rw table?          srv6-types:table-id
|           +--rw weight?         uint32
|           +--rw role?           enumeration
|           +--rw backup-path-index? uint8
|           +--rw sid-list
|               +--rw out-sid* [sid]
|                   +--rw sid      srv6-types:srv6-sid
+--rw end-x_osp
|   +--rw protected?    boolean
|   +--rw paths
|       +--rw path* [path-index]
|           +--rw path-index      uint8
|           +--rw interface?      if:interface-ref
|           +--rw next-hop?       inet:ipv6-address
|           +--rw table?          srv6-types:table-id
|           +--rw weight?         uint32
|           +--rw role?           enumeration
|           +--rw backup-path-index? uint8
|           +--rw sid-list
|               +--rw out-sid* [sid]
|                   +--rw sid      srv6-types:srv6-sid
+--rw end-x_psp_osp
|   +--rw protected?    boolean
|   +--rw paths
|       +--rw path* [path-index]
|           +--rw path-index      uint8
|           +--rw interface?      if:interface-ref
|           +--rw next-hop?       inet:ipv6-address
|           +--rw table?          srv6-types:table-id
|           +--rw weight?         uint32
|           +--rw role?           enumeration
|           +--rw backup-path-index? uint8
|           +--rw sid-list
|               +--rw out-sid* [sid]
|                   +--rw sid      srv6-types:srv6-sid
+--rw end-x_osp_osp
|   +--rw protected?    boolean
|   +--rw paths
|       +--rw path* [path-index]
|           +--rw path-index      uint8

```

```

        +--rw interface?          if:interface-ref
        +--rw next-hop?           inet:ipv6-address
        +--rw table?             srv6-types:table-id
        +--rw weight?            uint32
        +--rw role?              enumeration
        +--rw backup-path-index?  uint8
        +--rw sid-list
            +--rw out-sid* [sid]
            +--rw sid          srv6-types:srv6-sid
+--rw end-x_psp_usp_usd
+--rw protected?  boolean
+--rw paths
    +--rw path* [path-index]
        +--rw path-index          uint8
        +--rw interface?         if:interface-ref
        +--rw next-hop?          inet:ipv6-address
        +--rw table?            srv6-types:table-id
        +--rw weight?           uint32
        +--rw role?             enumeration
        +--rw backup-path-index? uint8
        +--rw sid-list
            +--rw out-sid* [sid]
            +--rw sid        srv6-types:srv6-sid
+--rw end-b6-insert
+--rw policy-name  string
+--rw paths
    +--rw path* [path-index]
        +--rw path-index          uint8
        +--rw interface?         if:interface-ref
        +--rw next-hop?          inet:ipv6-address
        +--rw table?            srv6-types:table-id
        +--rw weight?           uint32
        +--rw role?             enumeration
        +--rw backup-path-index? uint8
        +--rw sid-list
            +--rw out-sid* [sid]
            +--rw sid        srv6-types:srv6-sid
+--rw end-b6-encaps
+--rw policy-name  string
+--rw source-address  inet:ipv6-address
+--rw paths
    +--rw path* [path-index]
        +--rw path-index          uint8
        +--rw interface?         if:interface-ref
        +--rw next-hop?          inet:ipv6-address
        +--rw table?            srv6-types:table-id
        +--rw weight?           uint32
        +--rw role?             enumeration

```

```

        +---rw backup-path-index?    uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid          srv6-types:srv6-sid
+---rw end-bm
+---rw policy-name    string
+---rw paths
    +---rw path* [path-index]
        +---rw path-index            uint8
        +---rw interface?           if:interface-ref
        +---rw next-hop?            inet:ip-address
        +---rw weight?              uint32
        +---rw role?                enumeration
        +---rw backup-path-index?    uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid          srv6-types:srv6-sid
+---rw end-dx6
+---rw paths
    +---rw path* [path-index]
        +---rw path-index            uint8
        +---rw interface?           if:interface-ref
        +---rw next-hop?            inet:ipv6-address
        +---rw table?              srv6-types:table-id
        +---rw weight?              uint32
        +---rw role?                enumeration
        +---rw backup-path-index?    uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid          srv6-types:srv6-sid
+---rw end-dx4
+---rw paths
    +---rw path* [path-index]
        +---rw path-index            uint8
        +---rw interface?           if:interface-ref
        +---rw next-hop?            inet:ipv4-address
        +---rw table?              srv6-types:table-id
        +---rw weight?              uint32
        +---rw role?                enumeration
        +---rw backup-path-index?    uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid          srv6-types:srv6-sid
+---rw end-dt6
| +---rw lookup-table-ipv6    srv6-types:table-id
+---rw end-dt4
| +---rw lookup-table-ipv4    srv6-types:table-id
+---rw end-dt46

```

```

|   +--rw lookup-table-ipv4      srv6-types:table-id
|   +--rw lookup-table-ipv6      srv6-types:table-id
+--rw end-dx2
|   +--rw path
|       +--rw l2-interface      if:interface-ref
+--rw end-dx2v
|   +--rw lookup-table-vlan      srv6-types:table-id
+--rw end-dt2u
|   +--rw lookup-table-mac       srv6-types:table-id
+--rw end-dt2m
|   +--rw flooding-table         srv6-types:table-id
|   +--rw paths
|       +--rw path* [path-index]
|           +--rw path-index     uint8
|           +--rw l2-interface?  if:interface-ref
+--rw end-op
+--rw end-otp

```

Figure 4: SRv6 Static - Config Tree

3.4.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.4.1 can be retrieved from the same tree. The state regarding the local-SIDs created by SRv6-static model can be obtained using the state model of SRv6-base. Hence, there is no additional state identified at this time for SRv6-static.

3.4.3. Notification

None.

4. Pending Items

Following are the items that will be addressed in next revisions:

- o Align SRv6 base with SR (MPLS) model [I-D.ietf-spring-sr-yang].
- o Extend local-SID collision event/notification in SRv6-base model.
- o Add RPC support in the SRv6-base model.
- o Add ARGS support in the SRv6-Static model.
- o QoS support

5. YANG Specification

Following are actual YANG definition for SRv6 modules defined earlier in the document.

5.1. SRv6 Types

This YANG module imports types defined in [RFC6991].

Moreover, the module models behaviors defined in [I-D.ietf-spring-srv6-network-programming], [I-D.xuclad-spring-sr-service-chaining], and [I-D.ietf-dmm-srv6-mobile-uplane].

```
<CODE BEGINS> file "ietf-srv6-types@2019-10-30.yang" -->

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-types";
  prefix srv6-types;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF SPRING Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/spring/>
    WG List:  <mailto:spring@ietf.org>

    Editor:   Kamran Raza
              <mailto:skraza@cisco.com>

    Editor:   Jaganbabu Rajamanickam
              <mailto:jrajaman@cisco.com>

    Editor:   Xufeng Liu
              <mailto:xufeng.liu.ietf@gmail.com>

    Editor:   Zhibo Hu
              <mailto:huzhibo@huawei.com>
```

Editor: Iftekhar Hussain
<mailto:IHussain@infinera.com>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Daniel Voyer
<mailto:daniel.voyer@bell.ca>

Editor: Hani Elmalky
<mailto:hani.elmalky@ericsson.com>

Editor: Satoru Matsushima
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam
<mailto:ahmed.abdelsalam@gssi.it>

";

description

"This YANG module defines the essential types for the management of Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).";

reference "RFC XXXX";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

revision 2019-10-30 {

description

"Renaming of some types";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
}

revision 2019-07-08 {
  description
    "Alignment with latest SRv6 network programming";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

revision 2018-10-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

identity srv6-endpoint-type {
  description
    "Base identity from which specific SRv6 Endpoint types are
    derived.";
}

/* Endpoints defined under draft-ietf-spring-
 * srv6-network-programming */

identity End {
  base srv6-endpoint-type;
  description
    "End function (variant: no PSP, no USP).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_PSP {
  base srv6-endpoint-type;
  description
    "End function (variant: PSP only).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_USP {
```

```
    base srv6-endpoint-type;
    description
        "End function (variant: USP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_PSP_USP {
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: no PSP, no USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USP {
```

```
base srv6-endpoint-type;
description
    "Endpoint with cross-connect to an array
    of layer-3 adjacencies (variant: PSP and USP).";
reference
    "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.T {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: no PSP, no USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: USP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP and USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity End.B6.Insert {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SRv6 Policy";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.B6.Encaps {
  base srv6-endpoint-type;
  description
    "This is a variation of the End.B6.Insert behavior
     where the SRv6 Policy also includes an
     IPv6 Source Address A.";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.BM {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SR-MPLS Policy";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect
     to an array of IPv6 adjacencies";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect
     to an array of IPv4 adjacencies";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity End.DT6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
     IPv6 table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
     IPv4 table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT46 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific IP
     (IPv4 or IPv6) table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX2 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and Layer-2
     cross-connect to an L2 interface";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX2V {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
     VLAN L2 table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}
```

```
}

identity End.DT2U {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
    unicast MAC L2 table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT2M {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific L2 table
    flooding";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.OP {
  base srv6-endpoint-type;
  description
    "Endpoint for OAM operation of punt";
  reference
    "draft-ietf-6man-spring-srv6-oam-00";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.OTP {
  base srv6-endpoint-type;
  description
    "Endpoint for OAM operation of timestamp and punt";
  reference
    "draft-ietf-6man-spring-srv6-oam-00";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.S {
  base srv6-endpoint-type;
  description
    "Endpoint in search of a target in table TE";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}
```



```
identity End.B6.Insert.Red {
  base srv6-endpoint-type;
  description
    "This is a reduced insert variation of the End.B6.Insert
    behavior";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.B6.Encaps.Red {
  base srv6-endpoint-type;
  description
    "This is a reduced encap variation of the End.B6.Encap
    behavior.";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End_USD {
  base srv6-endpoint-type;
  description
    "End function (variant: USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.PSP_USD {
  base srv6-endpoint-type;
  description
    "End function (variant: PSP and USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.USP_USD {
  base srv6-endpoint-type;
  description
    "End function (variant: USP and USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.PSP_USP_USD {
```

```
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP and USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity End.T_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and Layer-2
    cross-connect to an L2 interface";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: PSP and USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_USP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: USP and USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: PSP and USP and USD).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

/* Endpoints defined under draft-xuclad-spring-sr-service-chaining */

identity End.AS {
  base srv6-endpoint-type;
  description
    "Service-Chaining Static proxy for inner type (Ethernet,
    IPv4 or IPv6)";
  reference
```

```
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AD {
    base srv6-endpoint-type;
    description
        "Service-Chaining Dynamic proxy for inner type (Ethernet,
        IPv4 or IPv6)";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.ASM {
    base srv6-endpoint-type;
    description
        "Service-Chaining Shared memory SR proxy for inner type
        (Ethernet, IPv4 or IPv6)";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AM {
    base srv6-endpoint-type;
    description
        "Service-Chaining Masquerading SR proxy";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

/* Endpoints defined under draft-ietf-dmm-srv6-mobile-uplane */

identity End.MAP {
    base srv6-endpoint-type;
    description
        "DMM End.MAP";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.D {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP6.D";
```

```
        reference
            "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.E {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP6.E";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP4.D {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP4.D";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP4.E {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP4.E";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.Limit {
    base srv6-endpoint-type;
    description
        "DMM End.Limit";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-transit-type {
    description
        "Base identity from which SRv6 transit rule types are derived.";
}

identity T {
    base srv6-transit-type;
```

```
    description
        "Transit rule T";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Insert {
    base srv6-transit-type;
    description
        "Transit rule T.Insert with insertion of an SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Insert.Red {
    base srv6-transit-type;
    description
        "Transit rule T.Insert.Red with reduced insertion of an
        SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Encaps {
    base srv6-transit-type;
    description
        "Transit rule T.Encaps with encapsulated of an SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Encaps.Red {
    base srv6-transit-type;
    description
        "Transit rule T.Encaps.Red with reduced encap of an
        SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Encaps.L2 {
    base srv6-transit-type;
    description
```

```
        "Transit rule T.Encaps.l2 on the received L2 frame";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity T.Encaps.L2.Red {
    base srv6-transit-type;
    description
        "Transit rule T.Encaps.L2.Red on the received L2 frame";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-security-type {
    description
        "Base identity from which SRv6 Security rule types are
        derived.";
}

identity SEC-1 {
    base srv6-security-type;
    description
        "Security rule SEC-1";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity SEC-2 {
    base srv6-security-type;
    description
        "Security rule SEC-2";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity SEC-3 {
    base srv6-security-type;
    description
        "Security rule SEC-3";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity srv6-counter-type {
  description
    "Base identity from which SRv6 counter types are derived.";
}

identity CNT-1 {
  base srv6-counter-type;
  description
    "Counter rule CNT-1";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity CNT-2 {
  base srv6-counter-type;
  description
    "Counter rule CNT-2";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity CNT-3 {
  base srv6-counter-type;
  description
    "Counter rule CNT-3";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

typedef srv6-sid {
  type inet:ipv6-prefix;
  description
    "This type defines a SID value in SRv6";
}

typedef srv6-func-value {
  type uint32;
  description
    "This is a typedef for SID's FUNC value";
}

typedef srv6-func-value-reserved-type {
  type enumeration {
    enum invalid { value 0; description "Invalid function value"; }
  }
}
```



```
        description "SRv6 SID's FUNC Reserved values";
    }

    typedef srv6-locator-len {
        type uint8 {
            range "32 .. 96";
        }
        description
            "This type defines an SRv6 locator len with range constraints";
    }

    typedef srv6-sid-pfxlen {
        type uint8 {
            range "33 .. 128";
        }
        default 128;
        description
            "This type defines a SID prefixlen with range constraints";
    }

    typedef sid-alloc-type {
        type enumeration {
            enum Dynamic {
                description
                    "SID allocated dynamically.";
            }
            enum Explicit {
                description
                    "SID allocated with explicit (static) value";
            }
        }
        description
            "Types of sid allocation used.";
    }

    identity srv6-sid-owner-type {
        description
            "Base identity from which SID owner types are derived.";
    }

    identity isis {
        base srv6-sid-owner-type;
        description "ISIS";
    }

    identity ospfv3 {
        base srv6-sid-owner-type;
        description "OSPFv3";
    }
```

```
}

identity bgp {
    base srv6-sid-owner-type;
    description "BGP";
}

identity evpn {
    base srv6-sid-owner-type;
    description "EVPN";
}

identity sr-policy {
    base srv6-sid-owner-type;
    description "SR Policy";
}

identity service-function {
    base srv6-sid-owner-type;
    description "SF";
}

typedef table-id {
    type uint32;
    description
        "Routing/switching/bridging/VLAN Table Id";
}

typedef srv6-status-type {
    type enumeration {
        enum up { value 1; description "State is Up"; }
        enum down { description "State is Down"; }
    }
    description
        "Status type";
}

typedef srv6-nexthop-type {
    type enumeration {
        enum ipv4 { value 1; description "IPv4 next-hop"; }
        enum ipv6 { description "IPv6 next-hop"; }
        enum mpls { description "MPLS next-hop"; }
        enum l2 { description "L2 next-hop"; }
    }
    description
        "Forwarding Next-hop type";
}
```

```
} // module
```

```
<CODE ENDS>
```

Figure 5: ietf-srv6-types.yang

5.2. SRv6 Base

This YANG module imports types defined in [RFC6991], [RFC8294], [RFC8343], and [RFC8349].

```
<CODE BEGINS> file "ietf-srv6-base@2019-10-30.yang" -->

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-base {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-base";
  prefix srv6;

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-routing {
    prefix "rt";
  }
}
```

```
reference
  "RFC 8349: A YANG Data Model for Routing Management
  (NMDA version)";
}

import ietf-srv6-types {
  prefix srv6-types;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

organization
  "IETF SPRING Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/spring/>
  WG List:    <mailto:spring@ietf.org>

  Editor:     Kamran Raza
              <mailto:skraza@cisco.com>

  Editor:     Jaganbabu Rajamanickam
              <mailto:jrajaman@cisco.com>

  Editor:     Xufeng Liu
              <mailto:Xufeng_Liu@jabil.com>

  Editor:     Zhibo Hu
              <mailto:huzhibo@huawei.com>

  Editor:     Iftekhar Hussain
              <mailto:IHussain@infinera.com>

  Editor:     Himanshu Shah
              <mailto:hshah@ciena.com>

  Editor:     Daniel Voyer
              <mailto:daniel.voyer@bell.ca>

  Editor:     Hani Elmalky
              <mailto:hani.elmalky@ericsson.com>

  Editor:     Satoru Matsushima
              <mailto:satoru.matsushima@gmail.com>

  Editor:     Katsuhiro Horiba
              <mailto:katsuhiro.horiba@g.softbank.co.jp>
```

Editor: Ahmed AbdelSalam
<mailto:ahmed.abdelsalam@gssi.it>

";

description

"This YANG module defines the essential elements for the management of Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2017 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).";

reference "RFC XXXX";

revision 2019-10-30 {

description

"Alignment with SRv6 network programming";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2019-07-08 {

description

"Alignment with SRv6 network programming";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2018-10-22 {

description

"Initial revision.";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

```
/*
 * Common
 */

grouping path-attrs-cmn {
  description
    "Path properties -common for v4/v6";

  leaf weight {
    type uint32;
    description
      "This value is used to compute a loadshare to perform un-equal
      load balancing when multiple outgoing path(s) are specified. A
      share is computed as a ratio of this number to the total under
      all configured path(s).";
  }

  leaf role {
    type enumeration {
      enum PRIMARY {
        description "Path as primary traffic carrying"; }
      enum BACKUP { description "Path acts as a backup"; }
      enum PRIMARY_AND_BACKUP {
        description "Path acts as primary and backup simultaneously"; }
    }

    description "The path role";
  }

  leaf backup-path-index {
    type uint8;
    description "Index of the protecting (backup) path";
  }
}

grouping path-out-sids {
  description "Grouping for path's SID stack";

  list out-sid {
    key "sid";
    description "Out SID";

    leaf sid {
      type srv6-types:srv6-sid;
      description "SID value";
    }
  }
}
```

```
grouping path-out-labels {
  description "Grouping for path's label stack";

  list out-label {
    key "label";
    description "Out label";

    leaf label {
      type rt-types:mpls-label;
      description "Label value";
    }
  }
}

/*
 * Config and State
 */

grouping srv6-encap {
  description "Grouping for encap param config.";

  container encapsulation {
    description "Configure encapsulation related parameters";
    leaf source-address {
      type inet:ipv6-address;
      description "Specify a source address (for T.Encap).
                  The address must locally exists and be routable";
    }
  }

  container hop-limit {
    description "Configure IPv6 header's Hop-limit options";
    leaf value {
      type uint8;
      default 64;
      description "Set encapsulating outer IPv6 header's Hoplimit
                  field to specified value when doing
                  encapsulation";
    }
  }

  leaf propagate {
    type boolean;
    default false;
    description "IP TTL/Hop-limit propagation from encapsulated
                packet to encapsulating outer IPv6 header's
                Hoplimit field. When configured on decapsulation
                side, this refers to propagating Hop-limit from
                outer IPv6 header to inner header after decap";
  }
}
```

```
    }

    container traffic-class {
      description "Configure IPv6 header's Traffic-class options";
      leaf value {
        type uint8;
        default 0;
        description "Set encapsulating outer IPv6 header's
                     Traffic-class field to specified value when
                     doing encapsulation";
      }

      leaf propagate {
        type boolean;
        default false;
        description "Propagate (or map) Traffic-class/CoS/PCP from
                     the incoming packet or L2 Ethernet frame being
                     encapsulated to the encapsulating IPv6 header's
                     Traffic-class field.";
      }
    }
  }
}

grouping srv6-locator-state {
  description "SRv6 grouping Locastateor ";

  leaf operational-status {
    type srv6-types:srv6-status-type;
    config false;
    description "Indicates whether locator state is UP";
  }

  leaf is-in-address-conflict {
    type boolean;
    config false;
    description "Indicates whether locator address conflicts with
                 some other IPv6 address on the box";
  }
}

grouping srv6-locators {
  description "SRv6 locator grouping";

  container locators {
    description "SRv6 locators";
  }
}
```



```
list locator {
  key "name";
  description "Configure a SRv6 locator";

  leaf name {
    type string;
    description "Locator name";
  }

  leaf enable {
    type boolean;
    default false;
    description "Enable a SRv6 locator";
  }

  container prefix {
    description "Specify locator prefix value";
    leaf address {
      type inet:ipv6-address;
      mandatory true;
      description "IPv6 address";
    }
    leaf length {
      type srv6-types:srv6-locator-len;
      mandatory true;
      description "Locator (prefix) length";
    }
  }

  leaf algorithm {
    type uint32 {
      range "128..255";
    }

    description "Algorithm Id (for Flex-Algo)";
  }

  uses srv6-locator-state;
}

grouping srv6-stats-in {
  description "Grouping for inbound stats";

  leaf in-pkts {
    type yang:counter64;
    description
```

```
        "A cumulative counter of the total number of packets
        received";
    }

    leaf in-octets {
        type yang:counter64;
        description
            "A cumulative counter of the total bytes received.";
    }
}

grouping srv6-stats-out {
    description "Grouping for inbound stats";

    leaf out-pkts {
        type yang:counter64;
        description
            "A cumulative counter of the total number of packets
            transmitted";
    }

    leaf out-octets {
        type yang:counter64;
        description
            "A cumulative counter of the total bytes transmitted.";
    }
}

grouping path-out-sids-choice {
    description "Grouping for Out-SID choices";
    choice encap-type {
        description "Out-SID encap-based choice";
        case srv6 {
            uses path-out-sids;
        }
        case mpls {
            uses path-out-labels;
        }
    }
}

grouping local-sid-fwd-state {
    description "SRv6 local-SID forwarding state grouping";

    container forwarding {
        description "SRv6 local-SID forwarding state";

        leaf is-installed {
```

```
    type boolean;
    description "Indicates whether SID is installed in forwarding";
}

leaf next-hop-type {
    type srv6-types:srv6-nexthop-type;
    description "Forwarding next-hop types";
}

container paths {
    when "../is-installed = 'true'" {
        description "This container is valid only when the
            local-SID is installed in forwarding";
    }

    list path {
        key path-index;
        description "The list of paths associated with the SID";

        leaf path-index {
            type uint8;
            description "Index of the path";
        }

        container l2 {
            when "../../next-hop-type = 'l2'" {
                description "This container is valid only for L2 type
                    of NHs";
            }

            leaf interface {
                type if:interface-ref;
                description "The outgoing Layer2 interface";
            }

            description "L2 information";
        }

        container l3 {
            when "../../next-hop-type != 'l2'" {
                description "This container is valid only for L3 type
                    of NHs";
            }

            leaf interface {
                type if:interface-ref;
                description "The outgoing Layer3 interface";
            }
        }
    }
}
```

```
        leaf next-hop {
            type inet:ip-address;
            description "The IP address of the next-hop";
        }

        uses path-attrs-cmn;

        description "L3 information";
    }
    uses path-out-sids-choice;
}

description "Forwarding paths";
}
}

grouping srv6-state-sid {
    description "SRv6 SID state grouping";

    container local-sids {
        config false;
        description "Local-SID state";

        container counters {
            description "SRv6 counters";
            container cnt-3 {
                description "Counts SRv6 traffic received/dropped on local
                prefix not instantiated as local-SID";
                uses srv6-stats-in;
            }
        }
    }

    list local-sid {
        key "sid";
        description "Per-localSID Counters";

        leaf sid {
            type srv6-types:srv6-sid;
            description "Local SID value";
        }

        uses srv6-locator;

        leaf is-reserved {
            type boolean;
            description "Set to true if SID comes from reserved pool";
        }
    }
}
```

```
    leaf end-behavior-type {
        type identityref {
            base srv6-types:srv6-endpoint-type;
        }
        description "Type of SRv6 end behavior.";
    }

    leaf alloc-type {
        type srv6-types:sid-alloc-type;
        description
            "Type of sid allocation.";
    }

    list owner {
        key "type instance";
        description "SID Owner clients";
        leaf type {
            type identityref {
                base srv6-types:srv6-sid-owner-type;
            }
            description "SID owner/client type";
        }
        leaf instance {
            type string;
            description "Client instance";
        }
        leaf is-winner {
            type boolean;
            description "Is this client/owner the winning in terms of
                forwarding";
        }
    }

    uses local-sid-fwd-state;

    container counters {
        description "SRv6 per local-SID counters";

        container cnt-1 {
            description "Counts SRv6 traffic received on local-SID
                prefix and processed successfully";
            uses srv6-stats-in;
        }
    }
}
```

```
grouping srv6-support-ends {
  description "SRv6 End behavior support grouping";

  list end-behavior {
    key "type";
    description "End behavior support";

    leaf type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "End behavior (End*) type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-support-transits {
  description "SRv6 Transit behavior support grouping";

  list transit-behavior {
    key "type";
    description "Transit behavior support";
    leaf type {
      type identityref {
        base srv6-types:srv6-transit-type;
      }
      description "Transit behavior (T*) type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-support-signaled {
  description "SRv6 signaled parameter support grouping";

  container signaled-parameters {
    description "SRv6 signaled parameter support";

    leaf max-sl {
```

```
    type uint8;
    //mandatory true;
    description "Maximum value of the SL field in the SRH of
                 a received packet before applying the function
                 associated with a SID";
}
leaf max-end-pop-srh {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs in the top SRH in an
                 SRH stack to which the router can apply
                 PSP or USP flavors";
}
leaf max-t_insert {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs that can be inserted as
                 part of the T.insert behavior";
}
leaf max-t_encap {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs that can be inserted as
                 part of the T.Encap behavior";
}
leaf max-end_d {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs in an SRH when applying
                 End.DX6 and End.DT6 functions";
}
}
}

grouping srv6-support-security-rules {
    description "SRv6 Security rules grouping";

    list security-rule {
        key "type";
        description "Security rule support";

        leaf type {
            type identityref {
                base srv6-types:srv6-security-type;
            }
            description "Security rule type";
        }
        leaf supported {
```

```
        type boolean;
        mandatory true;
        description "True if supported";
    }
}

grouping srv6-support-counters {
    description "SRv6 Counters grouping";

    list counters {
        key "type";
        description "SRv6 counter support";

        leaf type {
            type identityref {
                base srv6-types:srv6-counter-type;
            }
            description "Counter type";
        }
        leaf supported {
            type boolean;
            mandatory true;
            description "True if supported";
        }
    }
}

grouping srv6-state-capabilities {
    description "SRv6 node capabilities grouping";
    container node-capabilities {
        config false;
        description "Node's SRv6 capabilities";

        uses srv6-support-ends;
        uses srv6-support-transits;
        uses srv6-support-signaled;
        uses srv6-support-security-rules;
        uses srv6-support-counters;
    }
}

augment "/rt:routing" {
    description
        "This augments routing-instance configuration with
        segment-routing SRv6.";

    container srv6 {
```



```
description "Segment Routing with IPv6 dataplane";

/* config */
leaf enable {
    type boolean;
    default false;
    description "Enable SRv6";
}

uses srv6-encap;
uses srv6-locators;
uses srv6-state-capabilities;
uses srv6-state-sid;
}

/* Notifications */

grouping srv6-locator {
    description
        "An absolute reference to an SRv6 locator";
    leaf locator {
        type leafref {
            path "/rt:routing/srv6:srv6/srv6:locators/srv6:locator/srv6:name";
        }
        description
            "Reference to a SRv6 locator.";
    }
}

notification srv6-locator-status-event {
    description
        "Notification event for a change of SRv6 locator operational
        status.";
    leaf operational-status {
        type srv6-types:srv6-status-type;
        description "Operational status";
    }
    uses srv6-locator;
}

notification srv6-sid-collision-event {
    description
        "Notification event for an SRv6 SID collision - i.e., attempt
        to bind an already bound SID to a new context";
    leaf sid {
        type srv6-types:srv6-sid;
        description "SRv6 SID";
    }
}
```

```

    }
    container existing {
        description "Current assignment / bind";
        leaf end-behavior-type {
            type identityref {
                base srv6-types:srv6-endpoint-type;
            }
            description "End type";
        }
        // TODO: More
    }
    container requested {
        description "Requested assignment / bind";

        leaf end-behavior-type {
            type identityref {
                base srv6-types:srv6-endpoint-type;
            }
            description "End type";
        }
    }
}
} // module
<CODE ENDS>

```

Figure 6: ietf-srv6-base.yang

5.3. SRv6 Static

This YANG module imports types defined in [RFC6991], [RFC8343], and [RFC8349].

```

<CODE BEGINS> file "ietf-srv6-static@2019-10-30.yang" -->

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-static {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-static";
    prefix srv6-static;

    import ietf-interfaces {

```

```
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
}

import ietf-routing {
    prefix "rt";
    reference
        "RFC 8349: A YANG Data Model for Routing Management (NMDA
        version)";
}

import ietf-srv6-types {
    prefix srv6-types;
    reference "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

import ietf-srv6-base {
    prefix srv6;
    reference "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

organization
    "IETF SPRING Working Group";
contact
    "WG Web:    <http://tools.ietf.org/wg/spring/>
    WG List:    <mailto:spring@ietf.org>

    Editor:     Kamran Raza
                <mailto:skraza@cisco.com>

    Editor:     Jaganbabu Rajamanickam
                <mailto:jrajanaman@cisco.com>

    Editor:     Xufeng Liu
                <mailto:xufeng.liu.ietf@gmail.com>

    Editor:     Zhibo Hu
                <mailto:huzhibo@huawei.com>
```

Editor: Iftekhar Hussain
<mailto:IHussain@infinera.com>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Daniel Voyer
<mailto:daniel.voyer@bell.ca>

Editor: Hani Elmalky
<mailto:hani.elmalky@ericsson.com>

Editor: Satoru Matsushima
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam
<mailto:ahmed.abdelsalam@gssi.it>

";

description

"This YANG module defines the essential elements for the management of Static application for Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).";

reference "RFC XXXX";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

revision 2019-10-30 {

description

"Extended model for EVPN behaviors";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove

```
    // this note
}

revision 2019-07-08 {
  description
    "Alignment with SRv6 network programming";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

revision 2018-10-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

/*
 * Config and State
 */

grouping path-attrs-v6 {
  description
    "IPv6 Path properties";

  leaf interface {
    type if:interface-ref;
    description "The outgoing interface";
  }

  leaf next-hop {
    type inet:ipv6-address;
    description "The IP address of the next-hop";
  }

  leaf table {
    type srv6-types:table-id;
    description "The routing table associated with the next-hop";
  }

  uses srv6:path-attrs-cmn;
}
```

```
grouping path-attrs-v4 {
  description
    "IPv4 Path properties";

  leaf interface {
    type if:interface-ref;
    description "The outgoing interface";
  }

  leaf next-hop {
    type inet:ipv4-address;
    description "The IP address of the next-hop";
  }

  leaf table {
    type srv6-types:table-id;
    description "The routing table associated with the next-hop";
  }

  uses srv6:path-attrs-cmn;
}

grouping path-attrs-mpls {
  description
    "MPLS Path properties";

  leaf interface {
    type if:interface-ref;
    description "The outgoing interface";
  }

  leaf next-hop {
    type inet:ip-address;
    description "The IP address of the next-hop";
  }

  uses srv6:path-attrs-cmn;
}

grouping multi-paths-v6 {
  description "Multipath grouping";

  container paths {
    description "List of outgoing paths";
    list path {
      key path-index;
      description "The list of paths associated with the SID";
    }
  }
}
```

```
    leaf path-index {
        type uint8;
        description "Index of the path";
    }

    uses path-attrs-v6;
    container sid-list {
        description "SID-list associated with the path";
        uses srv6:path-out-sids;
    }
}

}

grouping multi-paths-v4 {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            uses path-attrs-v4;
            container sid-list {
                description "SID-list associated with the path";
                uses srv6:path-out-sids;
            }
        }
    }
}

grouping multi-paths-mpls {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
            }
        }
    }
}
```

```
        description "Index of the path";
    }

    uses path-attrs-mpls;
    container sid-list {
        description "SID-list associated with the path";
        uses srv6:path-out-sids;
    }
}

}

grouping multi-paths-v6-BUM {
    description
        "Multipath grouping for EVPN bridging BUM use case";

    container paths {
        description
            "List of outgoing paths for flooding";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            leaf l2-interface {
                type if:interface-ref;
                description "The outgoing L2 interface for flooding";
            }
        }
    }
}

grouping srv6-sid-config {
    description
        "Configuration parameters relating to SRv6 sid.";

    leaf function {
        type srv6-types:srv6-func-value;
        description
            "SRv6 function value.";
    }
    leaf end-behavior-type {
        type identityref {
            base srv6-types:srv6-endpoint-type;
        }
    }
}
```



```
    }
    mandatory true;
    description
      "Type of SRv6 end behavior.";
  }

  container end {
    when "../end-behavior-type = 'End'" {
      description
        "This container is valid only when the user chooses End
        behavior (variant: no PSP, no USP).";
    }
    description
      "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: no PSP, no USP)";
  }

  container end_psp {
    when "../end-behavior-type = 'End_PSP'" {
      description
        "This container is valid only when the user chooses End
        behavior (variant: PSP only).";
    }
    description
      "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: PSP only)";
  }

  container end_usp {
    when "../end-behavior-type = 'End_USP'" {
      description
        "This container is valid only when the user chooses End
        behavior (variant: USP only).";
    }
    description
      "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: USP only)";
  }
}
```

```
}

container end_psp_usp {
  when "../end-behavior-type = 'End_PSP_USP'" {
    description
      "This container is valid only when the user chooses End
        behavior (variant: PSP/USP).";
  }
  description
    "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: PSP/USP)";
}

container end_usd {
  when "../end-behavior-type = 'End_USD'" {
    description
      "This container is valid only when the user chooses End
        behavior (variant: USD only).";
  }
  description
    "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: USD)";
}

container end_psp_usd {
  when "../end-behavior-type = 'End_PSP_USD'" {
    description
      "This container is valid only when the user chooses End
        behavior (variant: PSP/USD).";
  }
  description
    "The Endpoint function is the most basic function.
      FIB lookup on updated DA and forward accordingly
      to the matched entry.
      This is the SRv6 instantiation of a Prefix SID
      (variant: PSP/USD)";
}

container end_usp_usd {
  when "../end-behavior-type = 'End_USP_USD'" {
```

```
        description
            "This container is valid only when the user chooses End
            behavior (variant: USP/USD).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: USP/USD)";
}

container end_psp_usp_usd {
    when "../end-behavior-type = 'End_PSP_USP_IUSD'" {
        description
            "This container is valid only when the user chooses End
            behavior (variant: PSP/USP/USD).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: PSP/USP/USD)";
}

container end-t {
    when "../end-behavior-type = 'End.T'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: no PSP, no USP).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: no PSP,
        no USP).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";
    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
```

```
        "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp {
    when "../end-behavior-type = 'End.T_PSP'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: PSP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: PSP only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.

        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_usp {
    when "../end-behavior-type = 'End.T_USP'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: USP only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
    }
}
```

```
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp_usp {
    when "../end-behavior-type = 'End.T_PSP_USP'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USP/PSP).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: USP/PSP).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_usd {
    when "../end-behavior-type = 'End.T_USD'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USD only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: USD only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
    }
}
```

```
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp_usd {
    when "../end-behavior-type = 'End.T_PSP_USD'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: PSP/USD only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: PSP/USD
        only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_usp_usd {
    when "../end-behavior-type = 'End.T_USP_USD'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USP/USD only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant:
        USP/USD only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
```

```
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp_usp_usd {
    when "../end-behavior-type = 'End.T_PSP_USP_USD'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant:
        PSP/USP/USD only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-x {
    when "../end-behavior-type = 'End.X'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: no USP/PSP)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: no USP/PSP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
    }
}
```

```
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_psp {
    when "../end-behavior-type = 'End.X_PSP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP only).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_usp {
    when "../end-behavior-type = 'End.X_USP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USP only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: USP only).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}
```



```
    }

    container end-x_psp_usp {
        when "../end-behavior-type = 'End.X_PSP_USP'" {
            description
                "This container is valid only when the user chooses
                End.X behavior (variant: PSP/USP)";
        }
        description
            "Endpoint with cross-connect to an array of
            layer-3 adjacencies (variant: PSP/USP).
            Forward to layer-3 adjacency bound to the SID S.
            The End.X function is required to express any
            traffic-engineering policy.";

        leaf protected {
            type boolean;
            default false;
            description "Is Adj-SID protected?";
        }

        uses multi-paths-v6;
    }

    container end-x_usd {
        when "../end-behavior-type = 'End.X_USD'" {
            description
                "This container is valid only when the user chooses
                End.X behavior (variant: USD only)";
        }
        description
            "Endpoint with cross-connect to an array of
            layer-3 adjacencies (variant: PSP/USP).
            Forward to layer-3 adjacency bound to the SID S.
            The End.X function is required to express any
            traffic-engineering policy.";

        leaf protected {
            type boolean;
            default false;
            description "Is Adj-SID protected?";
        }

        uses multi-paths-v6;
    }

    container end-x_psp_usd {
```

```
    when "../end-behavior-type = 'End.X_PSP_USD'" {
      description
        "This container is valid only when the user chooses
        End.X behavior (variant: PSP/USD only)";
    }
    description
      "Endpoint with cross-connect to an array of
      layer-3 adjacencies (variant: PSP/USP).
      Forward to layer-3 adjacency bound to the SID S.
      The End.X function is required to express any
      traffic-engineering policy.";

    leaf protected {
      type boolean;
      default false;
      description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
  }

  container end-x_osp_usd {
    when "../end-behavior-type = 'End.X_USP_USD'" {
      description
        "This container is valid only when the user chooses
        End.X behavior (variant: USP/USD only)";
    }
    description
      "Endpoint with cross-connect to an array of
      layer-3 adjacencies (variant: PSP/USP).
      Forward to layer-3 adjacency bound to the SID S.
      The End.X function is required to express any
      traffic-engineering policy.";

    leaf protected {
      type boolean;
      default false;
      description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
  }

  container end-x_osp_osp_usd {
    when "../end-behavior-type = 'End.X_PSP_USP_USD'" {
      description
        "This container is valid only when the user chooses
        End.X behavior (variant: PSP/USP/USD only)";
    }
```

```
}
description
  "Endpoint with cross-connect to an array of
  layer-3 adjacencies (variant: PSP/USP).
  Forward to layer-3 adjacency bound to the SID S.
  The End.X function is required to express any
  traffic-engineering policy.";

leaf protected {
  type boolean;
  default false;
  description "Is Adj-SID protected?";
}

uses multi-paths-v6;
}

container end-b6-insert {
  when "../end-behavior-type = 'End.B6.Insert' or
  ../end-behavior-type = 'End.B6.Insert.Red'" {
    description
      "This container is valid only when the user chooses
      End.B6.Insert or End.B6.Insert.Red behavior.";
  }
  description
    "Endpoint bound to an SRv6 Policy.
    Insert SRH based on the policy and forward the
    packet toward the first hop configured in the policy.
    This is the SRv6 instantiation of a Binding SID.";

  // TODO presence "Mandatory child only if container is present";

  leaf policy-name {
    type string;
    mandatory true;
    description "SRv6 policy name.";
  }

  uses multi-paths-v6;
}

container end-b6-encaps {
  when "../end-behavior-type = 'End.B6.Encaps' or
  ../end-behavior-type = 'End.B6.Encaps.Red'" {
    description
      "This container is valid only when the user chooses
      End.B6.Encaps or End.B6.Encaps.Red behavior.";
  }
}
```

```
description
  "This is a variation of the End.B6 behavior where
  the SRv6 Policy also includes an IPv6 Source
  Address.
  Insert SRH based on the policy and update the
  source IP and forward the packet toward the
  first hop configured in the policy.
  Instead of simply inserting an SRH with the
  policy (End.B6), this behavior also adds an
  outer IPv6 header.";

// TODO presence  "Mandatory child only if container is present";

leaf policy-name {
  type string;
  mandatory true;
  description "SRv6 policy name.";
}
leaf source-address {
  type inet:ipv6-address;
  mandatory true;
  description
    "IPv6 source address for Encap.";
}

uses multi-paths-v6;
}

container end-bm {
  when "../end-behavior-type = 'End.BM'" {
    description
      "This container is valid only when the user chooses
      End.BM behavior.";
  }

  description
    "Endpoint bound to an SR-MPLS Policy.
    push an MPLS label stack <L1, L2, L3> on the
    received packet and forward the according to
    Lable L1.
    This is an SRv6 instantiation of an SR-MPLS Binding SID.";

  // TODO presence  "Mandatory child only if container is present";

  leaf policy-name {
    type string;
    mandatory true;
    description "SRv6 policy name";
  }
}
```

```
    }
    uses multi-paths-mpls;
}

container end-dx6 {
  when "../end-behavior-type = 'End.DX6'" {
    description
      "This container is valid only when the user chooses
      End.DX6 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
    an array of IPv6 adjacencies. Pop the (outer)
    IPv6 header and its extension headers and forward
    to layer-3 adjacency bound to the SID S.
    The End.DX6 used in the L3VPN use-case.";

  uses multi-paths-v6;
  // TODO: Backup path of type "Lookup in table"
}

container end-dx4 {
  when "../end-behavior-type = 'End.DX4'" {
    description
      "This container is valid only when the user chooses
      End.DX4 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
    an array of IPv4 adjacencies.
    Pop the (outer) IPv6 header and its extension
    header and forward to layer-3 adjacency bound
    to the SID S.
    This would be equivalent to the per-CE VPN
    label in MPLS.";

  uses multi-paths-v4;
  // TODO: Backup path of type "Lookup in table"
}

container end-dt6 {
  when "../end-behavior-type = 'End.DT6'" {
    description
      "This container is valid only when the user chooses
      End.DT6 behavior.";
  }
  description
    "Endpoint with decapsulation and specific IPv6 table
    lookup."
```

```
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv6 DA in IPv6
        table T and forward via the matched table entry.
        End.DT6 function is used in L3VPN use-case.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv6 table";
    }
}
container end-dt4 {
    when "../end-behavior-type = 'End.DT4'" {
        description
            "This container is valid only when the user chooses
            End.DT4 behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        IPv4 table lookup.
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv4 DA in IPv4
        table T and forward via the matched table entry.
        This would be equivalent to the per-VRF VPN label
        in MPLS.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv4 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv4 table";
    }
}
container end-dt46 {
    when "../end-behavior-type = 'End.DT46'" {
        description
            "This container is valid only when the user chooses
            End.DT46 behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        IP table lookup.
        Depending on the protocol type (IPv4 or IPv6)
```

```
    of the inner ip packet and the specific VRF name
    forward the packet.
    This would be equivalent to the per-VRF VPN
    label in MPLS.";

    // TODO presence  "Mandatory child only if container is present";

    leaf lookup-table-ipv4 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv4 table";
    }
    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv6 table";
    }
}

/* EVPN END behavior types */
container end-dx2 {
    when "../end-behavior-type = 'End.DX2'" {
        description
            "This container is valid only when the user chooses
            End.DX2 behavior.";
    }
    description
        "This is an Endpoint with decapsulation and Layer-2
        cross-connect to OIF.
        Pop the (outer) IPv6 header and its extension headers.
        Forward the resulting frame via OIF associated to the SID.
        The End.DX2 function is the L2VPN/EVPN VPWS use-case.";

    container path {
        description "Outgoing path";
        leaf l2-interface {
            type if:interface-ref;
            mandatory true;
            description "Outgoing L2 interface";
        }
    }
}

container end-dx2v {
    when "../end-behavior-type = 'End.DX2V'" {
        description
            "This container is valid only when the user chooses
            End.DX2V behavior.";
```

```
    }
    description
      "Endpoint with decapsulation and specific VLAN
      L2 table lookup.
      Pop the (outer) IPv6 header and its extension headers.
      Lookup the exposed inner VLANs in L2 table T.
      Forward via the matched table entry.
      The End.DX2V is used for EVPN Flexible cross-connect
      use-cases";

    leaf lookup-table-vlan {
      type srv6-types:table-id;
      mandatory true;
      description
        "VLAN lookup table. There could be multiple
        vlan demux tables on the node, where a DX2V SID
        points to one vlan table";
    }
  }

  container end-dt2u {
    when "../end-behavior-type = 'End.DT2U'" {
      description
        "This container is valid only when the user chooses
        End.DT2U behavior.";
    }
    description
      "Endpoint with decapsulation and specific
      unicast L2 MAC table lookup.
      Pop the (outer) IPv6 header and its extension headers.
      Learn the exposed inner MAC SA in L2 MAC table T.
      Lookup the exposed inner MAC DA in L2 MAC table T.
      Forward via the matched T entry else to all L2OIF in T.
      The End.DT2U is used for EVPN Bridging unicast use cases";

    leaf lookup-table-mac {
      type srv6-types:table-id;
      mandatory true;
      description "MAC L2 lookup table";
    }
  }

  container end-dt2m {
    when "../end-behavior-type = 'End.DT2M'" {
      description
        "This container is valid only when the user chooses
        End.DT2M behavior.";
    }
  }
```



```
description
  "Endpoint with decapsulation and specific flooding table.
  Pop the (outer) IPv6 header and its extension headers.
  Learn the exposed inner MAC SA in L2 MAC table T.
  Forward on all L2OIF (in the flooding table) excluding the one
  identified by Arg.FE2.
  The End.DT2M is used for EVPN Bridging BUM use case with
  ESI (Split Horizon) filtering capability.";

leaf flooding-table {
  type srv6-types:table-id;
  mandatory true;
  description "L2 Flooding table (list of OIFs)";
}

uses multi-paths-v6-BUM;

/* TODO - Support for argument Arg.FE2. It is an argument specific
   to EVPN ESI filtering and EVPN-ETREE used to exclude specific
   OIF (or set of OIFs) from flooding table. */
}

/* End of EVPN END behavior types */

container end-op {
  when "../end-behavior-type = 'End.OP'" {
    description
      "This container is valid only when the user chooses
      End.OP behavior.";
  }
  description
    "Endpoint for OAM with punt behavior";
}

container end-otp {
  when "../end-behavior-type = 'End.OTP'" {
    description
      "This container is valid only when the user chooses
      End.OTP behavior.";
  }
  description
    "Endpoint for OAM with timestamp and punt behavior";
}
}

grouping srv6-static-cfg {
  description
    "Grouping configuration and operation for SRv6 sid.";
}
```

```
    list sid {
        key "function";
        description "List of locally instantiated SIDs";

        uses srv6-sid-config;
    }
}

augment "/rt:routing/srv6:srv6/srv6:locators/srv6:locator" {
    description
        "This augments locator leaf within SRv6.";

    container static {
        description "Static SRv6";

        /* Local SIDs */
        container local-sids {
            description
                "SRv6-static locally instantiated SIDs";

            uses srv6-static-cfg;
            /* no state for now; SID state accessible through base model */
        }
    }
}
} // module

<CODE ENDS>
```

Figure 7: ietf-srv6-static.yang

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

It goes without saying that this specification also inherits the security considerations captured in the SRv6 specification document [I-D.ietf-spring-srv6-network-programming].

7. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

| URI | Registrant | XML |
|--|------------|-----|
| urn:ietf:params:xml:ns:yang:ietf-srv6-types | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-srv6-base | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-srv6-static | The IESG | N/A |

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

| Name | Namespace | Prefix | Reference |
|------------------|--|-------------|---------------|
| ietf-srv6-types | urn:ietf:params:xml:ns:yang:ietf-srv6-types | srv6-types | This document |
| ietf-srv6-base | urn:ietf:params:xml:ns:yang:ietf-srv6-base | srv6 | This document |
| ietf-srv6-static | urn:ietf:params:xml:ns:yang:ietf-srv6-static | srv6-static | This document |

-- RFC Editor: Replace "This document" with the document RFC number at time of publication, and remove this note.

8. Acknowledgments

The authors would like to acknowledge Darren Dukes, Les Ginsberg, Ahmed Bashandy, Rajesh Venkateswaran, and Mike Mallin for their review of some of the contents in this draft.

9. References

9.1. Normative References

- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-05 (work in progress), October 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.

- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

9.2. Informative References

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.
- [I-D.ietf-dmm-srv6-mobile-uplane]
Matsushima, S., Filsfils, C., Kohno, M., Camarillo, P., Voyer, D., and C. Perkins, "Segment Routing IPv6 for Mobile User Plane", draft-ietf-dmm-srv6-mobile-uplane-06 (work in progress), September 2019.
- [I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Lindem, A., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-13 (work in progress), July 2019.
- [I-D.xuclad-spring-sr-service-chaining]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Segment Routing for Service Chaining", draft-xuclad-spring-sr-service-chaining-01 (work in progress), March 2018.

Authors' Addresses

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Sonal Agarwal
Cisco Systems
Email: agarwaso@cisco.com

Xufeng Liu
Volta Networks
Email: xufeng.liu.ietf@gmail.com

Zhibo Hu
Huawei Technologies
Email: huzhibo@huawei.com

Iftekhhar Hussain
Infinera Corporation
Email: IHussain@infinera.com

Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Hani Elmalky
Email: hani.elmalky@ericsson.com

Satoru Matsushima
SoftBank
Email: satoru.matsushima@g.softbank.co.jp

Katsuhiro Horiba
SoftBank
Email: katsuhiro.horiba@g.softbank.co.jp

Jaganbabu Rajamanickam
Cisco Systems
Email: jrajaman@cisco.com

Ahmed AbdelSalam
Cisco Systems
Email: ahabdels@cisco.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

K. Raza
S. Agarwal
Cisco Systems

X. Liu
Volta Networks

Z. Hu
Huawei Technologies

I. Hussain
Infinera Corporation

H. Shah
Ciena Corporation

D. Voyer
Bell Canada

S. Matsushima
K. Horiba
SoftBank

H. Elmalky

A. AbdelSalam
J. Rajamanickam
Cisco Systems

July 13, 2020

YANG Data Model for SRv6 Base and Static
draft-raza-spring-srv6-yang-06

Abstract

This document describes a YANG data model for Segment Routing IPv6 (SRv6) base. The model serves as a base framework for configuring and managing an SRv6 subsystem and expected to be augmented by other SRv6 technology models accordingly. Additionally, this document also specifies the model for the SRv6 Static application.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Specification of Requirements | 3 |
| 3. YANG Model | 4 |
| 3.1. Overview | 4 |
| 3.2. SRv6 Types | 4 |
| 3.3. SRv6 Base | 5 |
| 3.3.1. Configuration | 5 |
| 3.3.2. State | 6 |
| 3.3.3. Notification | 8 |
| 3.4. SRv6 Static | 9 |
| 3.4.1. Configuration | 9 |
| 3.4.2. State | 15 |
| 3.4.3. Notification | 15 |
| 4. Pending Items | 15 |
| 5. YANG Specification | 15 |
| 5.1. SRv6 Types | 15 |
| 5.2. SRv6 Base | 32 |
| 5.3. SRv6 Static | 48 |
| 6. Security Considerations | 71 |
| 7. IANA Considerations | 72 |
| 8. Acknowledgments | 72 |
| 9. References | 73 |
| 9.1. Normative References | 73 |
| 9.2. Informative References | 75 |

| | |
|--------------------|----|
| Authors' Addresses | 75 |
|--------------------|----|

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [RFC8402], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain. When applied to ipv6 data-plane (i.e. SRv6), SR requires a type of routing header (SRH) in an IPv6 packet that is used to encode an ordered list of IPv6 addresses (SIDs). The active segment is indicated by the Destination Address of the packet, and the next segment is indicated by a pointer in the SRH [RFC8754]. The various functions and behaviors corresponding to network programming using SRv6 are specified in [I-D.ietf-spring-srv6-network-programming].

This document introduces a YANG data model for base SRv6 that would serve as a base framework for configuring and managing an SRv6 subsystem. As needed, other SRv6 technology models (e.g. ISIS, OSPFv3, BGP, EVPN, Service Chaining) may augment this model. Furthermore, to illustrate basic behaviors as captured in [I-D.ietf-spring-srv6-network-programming], this document also specifies a YANG model for the SRv6-Static application.

The model currently defines the following constructs that are used for managing SRv6:

- o Configuration
- o Operational State
- o Notifications

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. YANG Model

3.1. Overview

This document defines following three new YANG modules:

- o ietf-srv6-types: defines common and basic types related to SRv6
- o ietf-srv6-base: specifies management model for SRv6 base constructs (locator, SIDs, etc.)
- o ietf-srv6-static: specifies management model for SRv6-static application

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

In this document, when a simplified graphical representation of YANG model is presented in a tree diagram, the meaning of the symbols in these tree diagrams is defined in [RFC8340].

3.2. SRv6 Types

SRv6 common types and definitions are defined in the new module "ietf-srv6-types". The main types defined in this module include:

- o srv6-sid: SRv6 SID
- o srv6-func-value: Typedef for FUNC value in an SRv6 SID
- o srv6-func-value-reserved-type: Enum (list) of "reserved" FUNC opcode
- o srv6-endpoint-type: SRv6 Endpoint behaviors [I-D.ietf-spring-srv6-network-programming] identity type
- o srv6-headend-type: SRv6 Headend behavior types [I-D.ietf-spring-srv6-network-programming] identity type
- o srv6-security-type: SRv6 Security rule type [I-D.ietf-spring-srv6-network-programming] identity type

- o `srv6-counter-type`: SRv6 Counter type
[I-D.ietf-spring-srv6-network-programming] identity type

The corresponding YANG specification for this module is captured in Section 5.1.

3.3. SRv6 Base

The base SRv6 model is specified in `ietf-srv6-base` module. This module augments `"/rt:routing:/sr:segment-routing"` [I-D.ietf-spring-sr-yang] and specifies the configuration, operational state, and notification events that are required to manage the base SRv6.

The corresponding YANG specification for this module is captured in Section 5.2.

3.3.1. Configuration

The module defines some fundamental items required to configure an SRv6 network:

- o `SRv6 Enablement`: Enable Segment-Routing SRv6 feature
- o `Encapsulation Parameters`: Provide encapsulation related parameters (such as `source-address`, `hop-limit`, and `traffic-class`) to be used when performing `T.Encap*` operation.
- o `Locator(s) Specification`: SRv6 locator is a fundamental construct for an SRv6 network. This is the construct from which SID (function values) are allocated that on the local box, and advertised to and used by remote nodes for reachability. A locator is identified by a name and has associated prefix and [IGP] algorithm. It can be configured as an anycast locator. It is possible to have more than one locator on a node (e.g. locator per algorithm, anycast and non-anycast locators, etc.).

Following is a simplified graphical tree representation of the data model for SRv6 base configuration

```

module: ietf-srv6-base
augment /rt:routing/sr:segment-routing:
  +--rw srv6
    +--rw enable?                boolean
    +--rw encapsulation
      +--rw source-address?      inet:ipv6-address
      +--rw hop-limit
        +--rw value?            uint8
        +--rw propagate?        boolean
      +--rw traffic-class
        +--rw value?            uint8
        +--rw propagate?        boolean
    +--rw locators
      +--rw locator* [name]
        +--rw name                string
        +--rw enable?            boolean
        +--rw prefix
          +--rw address          inet:ipv6-address
          +--rw length          srv6-types:srv6-locator-len
        +--rw algorithm?         uint32
        +--rw anycast?           boolean

```

Figure 1: SRv6 Base - Config Tree

3.3.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.3.1 can be retrieved from the same tree. This section defines other operational state items related to SRv6 base.

The operational state corresponding to the SRv6 base includes:

- o node capabilities: provides information on the node (hardware) capabilities and support regarding various SRv6 aspects and features including endpoint behaviors, headend behaviors, security rules, counter/stats support, and other SRv6 parameters that need to be signaled in an SRv6 network by the protocols.
- o locator: provides information related to a locator. The information includes locator operational state, and state of address conflict with any ipv6 address configured on local interfaces etc.
- o local-sid: provides information related to local-SIDs allocated and/or installed on the node. This includes two types of information:

1. aggregate across all local-SIDs such as aggregate counters
2. per local-SID information such as allocation type (dynamic or explicit), SID owner protocol(s)/client(s), forwarding [paths] information, and stats/counters.

Following is a simplified graphical tree representation of the data model for the SRv6 operational state (for read-only items):

```

module: ietf-srv6-base
augment /rt:routing/sr:segment-routing:
  +--rw srv6
    +--rw locators
      +--rw locator* [name]
        +--rw name string
        +--ro operational-status? srv6-types:srv6-status-type
        +--ro is-in-address-conflict? boolean
      +--ro node-capabilities
        +--ro end-behavior* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro headend-behavior* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro msd
          +--ro max-sl? uint8
          +--ro max-end-pop? uint8
          +--ro max-h_encap? uint8
          +--ro max-end_d? uint8
        +--ro security-rule* [type]
          +--ro type identityref
          +--ro supported boolean
        +--ro counters* [type]
          +--ro type identityref
          +--ro supported boolean
      +--ro local-sids
        +--ro counters
          +--ro cnt-3
            +--ro in-pkts? yang:counter64
            +--ro in-octets? yang:counter64
          +--ro local-sid* [sid]
            +--ro sid srv6-types:srv6-sid
            +--ro locator? -> /rt:routing/sr:segment-routing/srv6:sr
v6/locators/locator/name
          +--ro is-reserved? boolean
          +--ro end-behavior-type? identityref
          +--ro alloc-type? srv6-types:sid-alloc-type
          +--ro owner* [type instance]

```

```

|   +--ro type          identityref
|   +--ro instance      string
|   +--ro is-winner?    boolean
+--ro forwarding
|   +--ro is-installed?  boolean
|   +--ro next-hop-type?  srv6-types:srv6-nexthop-type
|   +--ro paths
|       +--ro path* [path-index]
|           +--ro path-index    uint8
|           +--ro l2
|               | +--ro interface?  if:interface-ref
|               +--ro l3
|                   +--ro interface?          if:interface-ref
|                   +--ro next-hop?          inet:ip-address
|                   +--ro weight?            uint32
|                   +--ro role?              enumeration
|                   +--ro backup-path-index?  uint8
|       +--ro (encap-type)?
|           +--:(srv6)
|               | +--ro out-sid* [sid]
|               | +--ro sid      srv6-types:srv6-sid
|           +--:(mpls)
|               +--ro out-label* [label]
|               +--ro label      rt-types:mpls-label
+--ro counters
    +--ro cnt-1
        +--ro in-pkts?      yang:counter64
        +--ro in-octets?    yang:counter64

```

Figure 2: SRv6 Base - State Tree

3.3.3. Notification

This model defines a list of notifications to inform an operator of important events detected during the SRv6 operation. These events include events related to:

- o locator operational state changes
- o local-SID collision event

Following is a simplified graphical tree representation of the data model for SRv6 notifications:


```

module: ietf-srv6-base

  notifications:
    +---n srv6-locator-status-event
    |   +---ro operational-status?   srv6-types:srv6-status-type
    |   +---ro locator?              -> /rt:routing/sr:segment-routing/srv6:srv6/lo
cators/locator/name
    +---n srv6-sid-collision-event
    |   +---ro sid?                  srv6-types:srv6-sid
    |   +---ro existing
    |   |   +---ro end-behavior-type?  identityref
    |   +---ro requested
    |   |   +---ro end-behavior-type?  identityref

```

Figure 3: SRv6 Base - Notification Tree

3.4. SRv6 Static

SRv6-Static application allows a user to specify SRv6 local SIDs and program them in the forwarding plane. The SRv6-Static model is captured in the ietf-srv6-static module.

The associated YANG specification for this module is captured in Section 5.3.

3.4.1. Configuration

The SRv6-Static configuration augments the SRv6-base locator tree `"/rt:routing/sr:segment-routing/srv6:srv6/locator"`

Following are salient features of the SRv6-Static config model:

- o Allows static (explicit) configuration for local-SIDs under a given locator.
- o Given that entry is scoped under a locator, the key for each entry is "function" value.
- o A user must also specify end-behavior type (End*) associated with the entry.
- o A user must also specify behavior-specific data with each entry. For example, for any end behavior requiring a table lookup, a lookup-table need be provided. Similarly, for any end behavior with forwarding next-hops need to specify next-hop information. The example of former include End, End.T, End.DT4, End.DT6, and End.DT46, whereas example of later include End.X, End.DX4, End.DX6, End.B6, End.BM etc.

- o Each local-SID entry has zero or more forwarding paths specified.
- o A forwarding path has next-hop type that depends on the end behavior, and could be either ipv6, or ipv4, or mpls, or l2 type. For example, End.X, End.DX4, End.DX6, End.B6, End.BM, and End.DX2 will have ipv6, ipv4, ipv6, ipv6, mpls, and l2 next-hop types respectively
- o For each forwarding next-hop type, the appropriate path attributes are to be specified as well. For L2 type, the only other information required is the L2 interface name. Whereas for L3 (ipv6, ipv4, mpls) types, the information includes L3 interface name, next-hop IP address, weight, and protection information.
- o Depending on the end behavior type, a forwarding path may have either MPLS or SRv6 encapsulation -- i.e., Stack of out-labels or Stack of SRv6 out-SIDs. The example of former is End.BM and example of later include the rest (End.X, End.DX4, End.DX6, End.B6 etc.).

Following is a simplified graphical tree representation of the data model for SRv6 Static configuration

```

module: ietf-srv6-static
augment /rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator:
  +--rw static
    +--rw local-sids
      +--rw sid* [function]
        +--rw function          srv6-types:srv6-func-value
        +--rw end-behavior-type identityref
        +--rw end
        +--rw end_psp
        +--rw end_usp
        +--rw end_psp_usp
        +--rw end_usd
        +--rw end_psp_usd
        +--rw end_usp_usd
        +--rw end_psp_usp_usd
        +--rw end-t
        | +--rw lookup-table-ipv6    srv6-types:table-id
        +--rw end-t_psp
        | +--rw lookup-table-ipv6    srv6-types:table-id
        +--rw end-t_usp
        | +--rw lookup-table-ipv6    srv6-types:table-id
        +--rw end-t_psp_usp
        | +--rw lookup-table-ipv6    srv6-types:table-id
        +--rw end-t_usd

```

```

|   +--rw lookup-table-ipv6      srv6-types:table-id
+--rw end-t_psp_usd
|   +--rw lookup-table-ipv6      srv6-types:table-id
+--rw end-t_usp_usd
|   +--rw lookup-table-ipv6      srv6-types:table-id
+--rw end-t_psp_usp_usd
|   +--rw lookup-table-ipv6      srv6-types:table-id
+--rw end-x
|   +--rw protected?    boolean
+--rw paths
|   +--rw path* [path-index]
|   |   +--rw path-index      uint8
|   |   +--rw interface?     if:interface-ref
|   |   +--rw next-hop?      inet:ipv6-address
|   |   +--rw table?         srv6-types:table-id
|   |   +--rw weight?        uint32
|   |   +--rw role?          enumeration
|   |   +--rw backup-path-index? uint8
|   |   +--rw sid-list
|   |   |   +--rw out-sid* [sid]
|   |   |   +--rw sid      srv6-types:srv6-sid
+--rw end-x_psp
|   +--rw protected?    boolean
+--rw paths
|   +--rw path* [path-index]
|   |   +--rw path-index      uint8
|   |   +--rw interface?     if:interface-ref
|   |   +--rw next-hop?      inet:ipv6-address
|   |   +--rw table?         srv6-types:table-id
|   |   +--rw weight?        uint32
|   |   +--rw role?          enumeration
|   |   +--rw backup-path-index? uint8
|   |   +--rw sid-list
|   |   |   +--rw out-sid* [sid]
|   |   |   +--rw sid      srv6-types:srv6-sid
+--rw end-x_usp
|   +--rw protected?    boolean
+--rw paths
|   +--rw path* [path-index]
|   |   +--rw path-index      uint8
|   |   +--rw interface?     if:interface-ref
|   |   +--rw next-hop?      inet:ipv6-address
|   |   +--rw table?         srv6-types:table-id
|   |   +--rw weight?        uint32
|   |   +--rw role?          enumeration
|   |   +--rw backup-path-index? uint8
|   |   +--rw sid-list
|   |   |   +--rw out-sid* [sid]

```

```

|           +---rw sid      srv6-types:srv6-sid
+---rw end-x_psp_usp
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?      inet:ipv6-address
|           +---rw table?         srv6-types:table-id
|           +---rw weight?        uint32
|           +---rw role?          enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid      srv6-types:srv6-sid
+---rw end-x_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?      inet:ipv6-address
|           +---rw table?         srv6-types:table-id
|           +---rw weight?        uint32
|           +---rw role?          enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid      srv6-types:srv6-sid
+---rw end-x_psp_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?      inet:ipv6-address
|           +---rw table?         srv6-types:table-id
|           +---rw weight?        uint32
|           +---rw role?          enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid      srv6-types:srv6-sid
+---rw end-x_usp_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8

```

```

    +---rw interface?          if:interface-ref
    +---rw next-hop?           inet:ipv6-address
    +---rw table?              srv6-types:table-id
    +---rw weight?             uint32
    +---rw role?               enumeration
    +---rw backup-path-index?  uint8
    +---rw sid-list
        +---rw out-sid* [sid]
        +---rw sid          srv6-types:srv6-sid
+---rw end-x_psp_usp_usd
+---rw protected?    boolean
+---rw paths
    +---rw path* [path-index]
        +---rw path-index          uint8
        +---rw interface?         if:interface-ref
        +---rw next-hop?          inet:ipv6-address
        +---rw table?             srv6-types:table-id
        +---rw weight?            uint32
        +---rw role?              enumeration
        +---rw backup-path-index? uint8
        +---rw sid-list
            +---rw out-sid* [sid]
            +---rw sid          srv6-types:srv6-sid
+---rw end-b6-encaps
+---rw policy-name      string
+---rw source-address   inet:ipv6-address
+---rw paths
    +---rw path* [path-index]
        +---rw path-index          uint8
        +---rw interface?         if:interface-ref
        +---rw next-hop?          inet:ipv6-address
        +---rw table?             srv6-types:table-id
        +---rw weight?            uint32
        +---rw role?              enumeration
        +---rw backup-path-index? uint8
        +---rw sid-list
            +---rw out-sid* [sid]
            +---rw sid          srv6-types:srv6-sid
+---rw end-bm
+---rw policy-name      string
+---rw paths
    +---rw path* [path-index]
        +---rw path-index          uint8
        +---rw interface?         if:interface-ref
        +---rw next-hop?          inet:ip-address
        +---rw weight?            uint32
        +---rw role?              enumeration
        +---rw backup-path-index? uint8

```

```

        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid      srv6-types:srv6-sid
+---rw end-dx6
+---rw paths
    +---rw path* [path-index]
        +---rw path-index      uint8
        +---rw interface?      if:interface-ref
        +---rw next-hop?       inet:ipv6-address
        +---rw table?          srv6-types:table-id
        +---rw weight?         uint32
        +---rw role?           enumeration
        +---rw backup-path-index? uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid      srv6-types:srv6-sid
+---rw end-dx4
+---rw paths
    +---rw path* [path-index]
        +---rw path-index      uint8
        +---rw interface?      if:interface-ref
        +---rw next-hop?       inet:ipv4-address
        +---rw table?          srv6-types:table-id
        +---rw weight?         uint32
        +---rw role?           enumeration
        +---rw backup-path-index? uint8
        +---rw sid-list
            +---rw out-sid* [sid]
                +---rw sid      srv6-types:srv6-sid
+---rw end-dt6
|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-dt4
|   +---rw lookup-table-ipv4      srv6-types:table-id
+---rw end-dt46
|   +---rw lookup-table-ipv4      srv6-types:table-id
|   +---rw lookup-table-ipv6      srv6-types:table-id
+---rw end-dx2
|   +---rw path
|       +---rw l2-interface      if:interface-ref
+---rw end-dx2v
|   +---rw lookup-table-vlan      srv6-types:table-id
+---rw end-dt2u
|   +---rw lookup-table-mac      srv6-types:table-id
+---rw end-dt2m
    +---rw flooding-table      srv6-types:table-id
    +---rw paths
        +---rw path* [path-index]
            +---rw path-index      uint8

```

```

+--rw l2-interface?   if:interface-ref

```

Figure 4: SRv6 Static - Config Tree

3.4.2. State

As per NMDA model, the state related to configuration items specified in above section Section 3.4.1 can be retrieved from the same tree. The state regarding the local-SIDs created by SRv6-static model can be obtained using the state model of SRv6-base. Hence, there is no additional state identified at this time for SRv6-static.

3.4.3. Notification

None.

4. Pending Items

Following are the items that will be addressed in next revisions:

- o Extend local-SID collision event/notification in SRv6-base model.
- o Add RPC support in the SRv6-base model.
- o Add ARGS support in the SRv6-Static model.
- o QoS support

5. YANG Specification

Following are actual YANG definition for SRv6 modules defined earlier in the document.

5.1. SRv6 Types

This YANG module imports types defined in [RFC6991].

Moreover, the module models behaviors defined in [I-D.ietf-spring-srv6-network-programming], [I-D.ietf-spring-sr-service-programming], and [I-D.ietf-dmm-srv6-mobile-uplane].

```

<CODE BEGINS> file "ietf-srv6-types@2020-07-13.yang" -->

```

```

// RFC Editor: replace the above date with the date of
// publication and remove this note.

```

```
module ietf-srv6-types {  
  yang-version 1.1;  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-types";  
  prefix srv6-types;  
  
  import ietf-inet-types {  
    prefix inet;  
    reference "RFC 6991: Common YANG Data Types";  
  }  
  
  organization  
    "IETF SPRING Working Group";  
  contact  
    "WG Web:    <http://tools.ietf.org/wg/spring/>  
    WG List:    <mailto:spring@ietf.org>  
  
    Editor:     Kamran Raza  
                <mailto:skraza@cisco.com>  
  
    Editor:     Jaganbabu Rajamanickam  
                <maito:jrajaman@cisco.com>  
  
    Editor:     Xufeng Liu  
                <mailto:xufeng.liu.ietf@gmail.com>  
  
    Editor:     Zhibo Hu  
                <mailto:huzhibo@huawei.com>  
  
    Editor:     Iftekhar Hussain  
                <mailto:IHussain@infinera.com>  
  
    Editor:     Himanshu Shah  
                <mailto:hshah@ciena.com>  
  
    Editor:     Daniel Voyer  
                <mailto:daniel.voyer@bell.ca>  
  
    Editor:     Hani Elmalky  
                <mailto:helmalky@google.com>  
  
    Editor:     Satoru Matsushima  
                <mailto:satoru.matsushima@gmail.com>  
  
    Editor:     Katsuhiko Horiba  
                <mailto:katsuhiko.horiba@g.softbank.co.jp>  
  
    Editor:     Ahmed AbdelSalam
```


<mailto:ahabdel@cisro.com>

”;

description

”This YANG module defines the essential types for the management of Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).”;

reference ”RFC XXXX”;

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

revision 2020-07-13 {

description

”Alignment with SRv6 net-pgm rev16”;

reference

”RFC XXXX: YANG Data Model for SRv6”;

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2019-10-30 {

description

”Renaming of some types”;

reference

”RFC XXXX: YANG Data Model for SRv6”;

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2019-07-08 {

description

”Alignment with latest SRv6 network programming”;

reference

”RFC XXXX: YANG Data Model for SRv6”;

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
}

revision 2018-10-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

identity srv6-endpoint-type {
  description
    "Base identity from which specific SRv6 Endpoint types are
    derived.";
}

/* Endpoints defined under draft-ietf-spring-
 * srv6-network-programming */

identity End {
  base srv6-endpoint-type;
  description
    "End function (variant: no PSP, no USP).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_PSP {
  base srv6-endpoint-type;
  description
    "End function (variant: PSP only).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_USP {
  base srv6-endpoint-type;
  description
    "End function (variant: USP only).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_PSP_USP {
```

```
base srv6-endpoint-type;
description
    "End function (variant: PSP and USP).";
reference
    "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.X {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
         of layer-3 adjacencies (variant: no PSP, no USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
         of layer-3 adjacencies (variant: PSP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
         of layer-3 adjacencies (variant: USP only).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
         of layer-3 adjacencies (variant: PSP and USP).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
// RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity End.T {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: no PSP, no USP).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: PSP only).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_USP {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: USP only).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USP {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
    (variant: PSP and USP).";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.B6.Encaps {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SRv6 Policy
    where the SRv6 Policy also includes an
    IPv6 Source Address A.";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
```

```
// RFC Editor: replace with actual RFC number and remove this note
}

identity End.BM {
  base srv6-endpoint-type;
  description
    "Endpoint bound to an SR-MPLS Policy";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect
    to an array of IPv6 adjacencies";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and cross-connect
    to an array of IPv4 adjacencies";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT6 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
    IPv6 table lookup";
  reference
    "draft-ietf-spring-srv6-network-programming-01";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT4 {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and specific
    IPv4 table lookup";
  reference
```

```
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT46 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific IP
        (IPv4 or IPv6) table lookup";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX2 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and Layer-2
        cross-connect to an L2 interface";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DX2V {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific
        VLAN L2 table lookup";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT2U {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific
        unicast MAC L2 table lookup";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.DT2M {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific L2 table
```

```
        flooding";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.S {
    base srv6-endpoint-type;
    description
        "Endpoint in search of a target in table TE";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.B6.Encaps.Red {
    base srv6-endpoint-type;
    description
        "This is a reduced encaps variation of the End.B6.Encap
        behavior.";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.PSP_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.USP_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: USP and USD).";
    reference
```

```
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.PSP_USP_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.X_PSP_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP and USP and USD).";
```



```
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and Layer-2
        cross-connect to an L2 interface";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.T_PSP_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP and USP and USD).";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

/* Endpoints defined under draft-xuclad-spring-sr-service-chaining */

identity End.AS {
```

```
    base srv6-endpoint-type;
    description
        "Service-Chaining Static proxy for inner type (Ethernet,
        IPv4 or IPv6)";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AD {
    base srv6-endpoint-type;
    description
        "Service-Chaining Dynamic proxy for inner type (Ethernet,
        IPv4 or IPv6)";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.ASM {
    base srv6-endpoint-type;
    description
        "Service-Chaining Shared memory SR proxy for inner type
        (Ethernet, IPv4 or IPv6)";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AM {
    base srv6-endpoint-type;
    description
        "Service-Chaining Masquerading SR proxy";
    reference
        "draft-xuclad-spring-sr-service-chaining-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

/* Endpoints defined under draft-ietf-dmm-srv6-mobile-uplane */

identity End.MAP {
    base srv6-endpoint-type;
    description
        "DMM End.MAP";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}
```

```
identity End.M.GTP6.D {
  base srv6-endpoint-type;
  description
    "DMM End.M.GTP6.D";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-05";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.E {
  base srv6-endpoint-type;
  description
    "DMM End.M.GTP6.E";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-05";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP4.D {
  base srv6-endpoint-type;
  description
    "DMM End.M.GTP4.D";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-05";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP4.E {
  base srv6-endpoint-type;
  description
    "DMM End.M.GTP4.E";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-05";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity End.Limit {
  base srv6-endpoint-type;
  description
    "DMM End.Limit";
  reference
    "draft-ietf-dmm-srv6-mobile-uplane-05";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-headend-type {
  description
    "Base identity from which SRv6 headend rule types are derived.";
```

```
}

identity H.Encaps {
  base srv6-headend-type;
  description
    "Headend rule H.Encaps with encapsulated of an SRv6 policy";
  reference
    "draft-ietf-spring-srv6-network-programming-16";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity H.Encaps.Red {
  base srv6-headend-type;
  description
    "Headend rule H.Encaps.Red with reduced encap of an
    SRv6 policy";
  reference
    "draft-ietf-spring-srv6-network-programming-16";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity H.Encaps.L2 {
  base srv6-headend-type;
  description
    "Headend rule H.Encaps.l2 on the received L2 frame";
  reference
    "draft-ietf-spring-srv6-network-programming-16";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity H.Encaps.L2.Red {
  base srv6-headend-type;
  description
    "Headend rule H.Encaps.L2.Red on the received L2 frame";
  reference
    "draft-ietf-spring-srv6-network-programming-16";
  // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-security-type {
  description
    "Base identity from which SRv6 Security rule types are
    derived.";
}

identity SEC-1 {
  base srv6-security-type;
  description
```

```
        "Security rule SEC-1";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity SEC-2 {
    base srv6-security-type;
    description
        "Security rule SEC-2";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity SEC-3 {
    base srv6-security-type;
    description
        "Security rule SEC-3";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-counter-type {
    description
        "Base identity from which SRv6 counter types are derived.";
}

identity CNT-1 {
    base srv6-counter-type;
    description
        "Counter rule CNT-1";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity CNT-2 {
    base srv6-counter-type;
    description
        "Counter rule CNT-2";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity CNT-3 {
```

```
    base srv6-counter-type;
    description
        "Counter rule CNT-3";
    reference
        "draft-ietf-spring-srv6-network-programming-01";
    // RFC Editor: replace with actual RFC number and remove this note
}

typedef srv6-sid {
    type inet:ipv6-prefix;
    description
        "This type defines a SID value in SRv6";
}

typedef srv6-func-value {
    type uint32;
    description
        "This is a typedef for SID's FUNC value";
}

typedef srv6-func-value-reserved-type {
    type enumeration {
        enum invalid { value 0; description "Invalid function value"; }
    }

    description "SRv6 SID's FUNC Reserved values";
}

typedef srv6-locator-len {
    type uint8 {
        range "32 .. 96";
    }
    description
        "This type defines an SRv6 locator len with range constraints";
}

typedef srv6-sid-pfxlen {
    type uint8 {
        range "32 .. 128";
    }
    default 128;
    description
        "This type defines a SID prefixlen with range constraints";
}

typedef sid-alloc-type {
    type enumeration {
        enum Dynamic {
```

```
        description
            "SID allocated dynamically.";
    }
    enum Explicit {
        description
            "SID allocated with explicit (static) value";
    }
}
description
    "Types of sid allocation used.";
}

identity srv6-sid-owner-type {
    description
        "Base identity from which SID owner types are derived.";
}

identity isis {
    base srv6-sid-owner-type;
    description "ISIS";
}

identity ospfv3 {
    base srv6-sid-owner-type;
    description "OSPFv3";
}

identity bgp {
    base srv6-sid-owner-type;
    description "BGP";
}

identity evpn {
    base srv6-sid-owner-type;
    description "EVPN";
}

identity sr-policy {
    base srv6-sid-owner-type;
    description "SR Policy";
}

identity service-function {
    base srv6-sid-owner-type;
    description "SF";
}

typedef table-id {
```

```

    type uint32;
    description
        "Routing/switching/bridging/VLAN Table Id";
}

typedef srv6-status-type {
    type enumeration {
        enum up { value 1; description "State is Up"; }
        enum down { description "State is Down"; }
    }
    description
        "Status type";
}

typedef srv6-nexthop-type {
    type enumeration {
        enum ipv4 { value 1; description "IPv4 next-hop"; }
        enum ipv6 { description "IPv6 next-hop"; }
        enum mpls { description "MPLS next-hop"; }
        enum l2 { description "L2 next-hop"; }
    }
    description
        "Forwarding Next-hop type";
}

} // module

<CODE ENDS>

```

Figure 5: ietf-srv6-types.yang

5.2. SRv6 Base

This YANG module imports types defined in [RFC6991], [RFC8294], [RFC8343], and [RFC8349].

```

<CODE BEGINS> file "ietf-srv6-base@2020-07-13.yang" -->

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-base {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-base";

```



```
prefix srv6;

import ietf-interfaces {
  prefix "if";
  reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-inet-types {
  prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-yang-types {
  prefix "yang";
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-routing-types {
  prefix "rt-types";
  reference "RFC 8294: Common YANG Data Types for the Routing Area";
}

import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

import ietf-segment-routing {
  prefix sr;
  reference "draft-ietf-spring-sr-yang";
}

import ietf-srv6-types {
  prefix srv6-types;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

organization
  "IETF SPRING Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/spring/>
  WG List:    <mailto:spring@ietf.org>

  Editor:     Kamran Raza
```

<mailto:skraza@cisco.com>

Editor: Jaganbabu Rajamanickam
<mailto:jrajaman@cisco.com>

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

Editor: Zhibo Hu
<mailto:huzhibo@huawei.com>

Editor: Iftekhar Hussain
<mailto:IHussain@infinera.com>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Daniel Voyer
<mailto:daniel.voyer@bell.ca>

Editor: Hani Elmalky
<mailto:helmalky@google.com>

Editor: Satoru Matsushima
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam
<mailto:ahabdel@cisco.com>

";

description

"This YANG module defines the essential elements for the management of Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2017 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).";

```
reference "RFC XXXX";

revision 2020-07-13 {
  description
    "Alignment with SRv6 network programming rev16";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

revision 2019-10-30 {
  description
    "Alignment with SRv6 network programming";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

revision 2019-07-08 {
  description
    "Alignment with SRv6 network programming";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

revision 2018-10-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

/*
 * Common
 */

grouping path-attrs-cmn {
  description
    "Path properties -common for v4/v6";

  leaf weight {
    type uint32;
```

```
    description
      "This value is used to compute a loadshare to perform un-equal
      load balancing when multiple outgoing path(s) are specified. A
      share is computed as a ratio of this number to the total under
      all configured path(s).";
  }

  leaf role {
    type enumeration {
      enum PRIMARY { description "Path as primary traffic carrying"; }
      enum BACKUP { description "Path acts as a backup"; }
      enum PRIMARY_AND_BACKUP {
        description "Path acts as primary and backup simultaneously"; }
    }
    description "The path role";
  }

  leaf backup-path-index {
    type uint8;
    description "Index of the protecting (backup) path";
  }
}

grouping path-out-sids {
  description "Grouping for path's SID stack";

  list out-sid {
    key "sid";
    description "Out SID";

    leaf sid {
      type srv6-types:srv6-sid;
      description "SID value";
    }
  }
}

grouping path-out-labels {
  description "Grouping for path's label stack";

  list out-label {
    key "label";
    description "Out label";

    leaf label {
      type rt-types:mpls-label;
      description "Label value";
    }
  }
}
```

```
    }

}

/*
 * Config and State
 */

grouping srv6-encap {
  description "Grouping for encap param config.";

  container encapsulation {
    description "Configure encapsulation related parameters";
    leaf source-address {
      type inet:ipv6-address;
      description "Specify a source address (for T.Encap).
                   The address must locally exists and be routable";
    }
    container hop-limit {
      description "Configure IPv6 header's Hop-limit options";
      leaf value {
        type uint8;
        default 64;
        description "Set encapsulating outer IPv6 header's Hoplimit
                     field to specified value when doing
                     encapsulation";
      }
    }

    leaf propagate {
      type boolean;
      default false;
      description "IP TTL/Hop-limit propagation from encapsulated
                   packet to encapsulating outer IPv6 header's
                   Hoplimit field. When configured on decapsulation
                   side, this refers to propagating Hop-limit from
                   outer IPv6 header to inner header after decap";
    }
  }

  container traffic-class {
    description "Configure IPv6 header's Traffic-class options";
    leaf value {
      type uint8;
      default 0;
      description "Set encapsulating outer IPv6 header's
                   Traffic-class field to specified value when
                   doing encapsulation";
    }
  }
}
```

```
    leaf propagate {
      type boolean;
      default false;
      description "Propagate (or map) Traffic-class/CoS/PCP from
                  the incoming packet or L2 Ethernet frame being
                  encapsulated to the encapsulating IPv6 header's
                  Traffic-class field.";
    }
  }
}

grouping srv6-locator-state {
  description "SRv6 grouping Locator state";

  leaf operational-status {
    type srv6-types:srv6-status-type;
    config false;
    description "Indicates whether locator state is UP";
  }

  leaf is-in-address-conflict {
    type boolean;
    config false;
    description "Indicates whether locator address conflicts with
                some other IPv6 address on the box";
  }
}

grouping srv6-locators {
  description "SRv6 locator grouping";

  container locators {
    description "SRv6 locators";

    list locator {
      key "name";
      description "Configure a SRv6 locator";

      leaf name {
        type string;
        description "Locator name";
      }

      leaf enable {
        type boolean;
        default false;
      }
    }
  }
}
```

```
    description "Enable a SRv6 locator";
  }

  container prefix {
    description "Specify locator prefix value";
    leaf address {
      type inet:ipv6-address;
      mandatory true;
      description "IPv6 address";
    }
    leaf length {
      type srv6-types:srv6-locator-len;
      mandatory true;
      description "Locator (prefix) length";
    }
  }

  leaf algorithm {
    type uint32 {
      range "128..255";
    }

    description "Algorithm Id (for Flex-Algo)";
  }

  leaf anycast {
    type boolean;
    default false;
    description "Set to true if locator is an Anycast locator";
  }

  uses srv6-locator-state;
}

}

grouping srv6-stats-in {
  description "Grouping for inbound stats";

  leaf in-pkts {
    type yang:counter64;
    description
      "A cumulative counter of the total number of packets
      received";
  }

  leaf in-octets {
    type yang:counter64;
  }
}
```

```
        description
            "A cumulative counter of the total bytes received.";
    }
}

grouping srv6-stats-out {
    description "Grouping for inbound stats";

    leaf out-pkts {
        type yang:counter64;
        description
            "A cumulative counter of the total number of packets
            transmitted";
    }

    leaf out-octets {
        type yang:counter64;
        description
            "A cumulative counter of the total bytes transmitted.";
    }
}

grouping path-out-sids-choice {
    description "Grouping for Out-SID choices";
    choice encap-type {
        description "Out-SID encap-based choice";
        case srv6 {
            uses path-out-sids;
        }
        case mpls {
            uses path-out-labels;
        }
    }
}

grouping local-sid-fwd-state {
    description "SRv6 local-SID forwarding state grouping";

    container forwarding {
        description "SRv6 local-SID forwarding state";

        leaf is-installed {
            type boolean;
            description "Indicates whether SID is installed in forwarding";
        }

        leaf next-hop-type {
            type srv6-types:srv6-nexthop-type;
        }
    }
}
```



```
    description "Forwarding next-hop types";
  }

  container paths {
    when "../is-installed = 'true'" {
      description "This container is valid only when the
        local-SID is installed in forwarding";
    }

    list path {
      key path-index;
      description "The list of paths associated with the SID";

      leaf path-index {
        type uint8;
        description "Index of the path";
      }

      container l2 {
        when "../../../next-hop-type = 'l2'" {
          description "This container is valid only for L2 type
            of NHs";
        }

        leaf interface {
          type if:interface-ref;
          description "The outgoing Layer2 interface";
        }

        description "L2 information";
      }

      container l3 {
        when "../../../next-hop-type != 'l2'" {
          description "This container is valid only for L3 type
            of NHs";
        }

        leaf interface {
          type if:interface-ref;
          description "The outgoing Layer3 interface";
        }

        leaf next-hop {
          type inet:ip-address;
          description "The IP address of the next-hop";
        }
      }
    }
  }
}
```

```
        uses path-attrs-cmn;

        description "L3 information";
    }
    uses path-out-sids-choice;
}

description "Forwarding paths";
}
}

grouping srv6-state-sid {
    description "SRv6 SID state grouping";

    container local-sids {
        config false;
        description "Local-SID state";

        container counters {
            description "SRv6 counters";
            container cnt-3 {
                description "Counts SRv6 traffic received/dropped on local
                prefix not instantiated as local-SID";
                uses srv6-stats-in;
            }
        }

        list local-sid {
            key "sid";
            description "Per-localSID Counters";

            leaf sid {
                type srv6-types:srv6-sid;
                description "Local SID value";
            }

            uses srv6-locator;

            leaf is-reserved {
                type boolean;
                description "Set to true if SID comes from reserved pool";
            }

            leaf end-behavior-type {
                type identityref {
                    base srv6-types:srv6-endpoint-type;
                }
            }
        }
    }
}
```

```
        description "Type of SRv6 end behavior.";
    }

    leaf alloc-type {
        type srv6-types:sid-alloc-type;
        description
            "Type of sid allocation.";
    }

    list owner {
        key "type instance";
        description "SID Owner clients";
        leaf type {
            type identityref {
                base srv6-types:srv6-sid-owner-type;
            }
            description "SID owner/client type";
        }
        leaf instance {
            type string;
            description "Client instance";
        }
        leaf is-winner {
            type boolean;
            description "Is this client/owner the winning in terms of
                forwarding";
        }
    }

    uses local-sid-fwd-state;

    container counters {
        description "SRv6 per local-SID counters";

        container cnt-1 {
            description "Counts SRv6 traffic received on local-SID
                prefix and processed successfully";
            uses srv6-stats-in;
        }
    }
}
}
}

grouping srv6-support-ends {
    description "SRv6 End behavior support grouping";

    list end-behavior {
```

```
    key "type";
    description "End behavior support";

    leaf type {
        type identityref {
            base srv6-types:srv6-endpoint-type;
        }
        description "End behavior (End*) type";
    }
    leaf supported {
        type boolean;
        mandatory true;
        description "True if supported";
    }
}

grouping srv6-support-headends {
    description "SRv6 Headend behavior support grouping";

    list headend-behavior {
        key "type";
        description "Headend behavior support";
        leaf type {
            type identityref {
                base srv6-types:srv6-headend-type;
            }
            description "Headend behavior (H*) type";
        }
        leaf supported {
            type boolean;
            mandatory true;
            description "True if supported";
        }
    }
}

grouping srv6-msd-signaled {
    description "SRv6 MSD signaled parameter support grouping";

    container msd {
        description "SRv6 signaled MSD parameter support";

        leaf max-sl {
            type uint8;
            description "Maximum value of the SL field in the SRH of
                a received packet before applying the Endpoint behavior
                associated with a SID";
        }
    }
}
```

```
    }
    leaf max-end-pop {
      type uint8;
      description "Maximum number of SIDs in the top SRH in an
                   SRH stack to which the router can apply
                   PSP or USP flavors";
    }
    leaf max-h_encap {
      type uint8;
      description "Maximum number of SIDs that can be pushed as
                   part of the H.Encaps* behavior";
    }
    leaf max-end_d {
      type uint8;
      description "Maximum number of SIDs in an SRH when applying
                   End.D* behaviors (e.g. End.X6 and End.DT6)";
    }
  }
}

grouping srv6-support-security-rules {
  description "SRv6 Security rules grouping";

  list security-rule {
    key "type";
    description "Security rule support";

    leaf type {
      type identityref {
        base srv6-types:srv6-security-type;
      }
      description "Security rule type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-support-counters {
  description "SRv6 Counters grouping";

  list counters {
    key "type";
    description "SRv6 counter support";
  }
}
```

```
    leaf type {
      type identityref {
        base srv6-types:srv6-counter-type;
      }
      description "Counter type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-state-capabilities {
  description "SRv6 node capabilities grouping";
  container node-capabilities {
    config false;
    description "Node's SRv6 capabilities";

    uses srv6-support-ends;
    uses srv6-support-headends;
    uses srv6-msd-signaled;
    uses srv6-support-security-rules;
    uses srv6-support-counters;
  }
}

augment "/rt:routing/sr:segment-routing" {
  description
    "This augments Segment Routing (SR) with SRv6.";

  container srv6 {
    description "Segment Routing with IPv6 dataplane";

    /* config */
    leaf enable {
      type boolean;
      default false;
      description "Enable SRv6";
    }

    uses srv6-encap;
    uses srv6-locators;
    uses srv6-state-capabilities;
    uses srv6-state-sid;
  }
}
```

```
/* Notifications */

grouping srv6-locator {
  description
    "An absolute reference to an SRv6 locator";
  leaf locator {
    type leafref {
      path "/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator
/srv6:name";
    }
    description
      "Reference to a SRv6 locator.";
  }
}

notification srv6-locator-status-event {
  description
    "Notification event for a change of SRv6 locator operational
status.";
  leaf operational-status {
    type srv6-types:srv6-status-type;
    description "Operational status";
  }
  uses srv6-locator;
}

notification srv6-sid-collision-event {
  description
    "Notification event for an SRv6 SID collision - i.e., attempt
to bind an already bound SID to a new context";
  leaf sid {
    type srv6-types:srv6-sid;
    description "SRv6 SID";
  }
  container existing {
    description "Current assignment / bind";
    leaf end-behavior-type {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "End type";
    }
    // TODO: More
  }
  container requested {
    description "Requested assignment / bind";

    leaf end-behavior-type {
      type identityref {
```

```
        base srv6-types:srv6-endpoint-type;
    }
    description "End type";
}
}
} // module
<CODE ENDS>
```

Figure 6: ietf-srv6-base.yang

5.3. SRv6 Static

This YANG module imports types defined in [RFC6991], [RFC8343], and [RFC8349].

```
<CODE BEGINS> file "ietf-srv6-static@2020-07-13.yang" -->

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-static {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-static";
    prefix srv6-static;

    import ietf-interfaces {
        prefix "if";
        reference "RFC 8343: A YANG Data Model for Interface Management";
    }

    import ietf-inet-types {
        prefix inet;
        reference "RFC 6991: Common YANG Data Types";
    }

    import ietf-routing {
        prefix "rt";
        reference
            "RFC 8349: A YANG Data Model for Routing Management (NMDA
            version)";
    }
}
```



```
import ietf-segment-routing {
  prefix sr;
  reference "draft-ietf-spring-sr-yang";
}

import ietf-srv6-types {
  prefix srv6-types;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

import ietf-srv6-base {
  prefix srv6;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

organization
  "IETF SPRING Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/spring/>
   WG List:   <mailto:spring@ietf.org>

   Editor:    Kamran Raza
              <mailto:skraza@cisco.com>

   Editor:    Jaganbabu Rajamanickam
              <maito:jrajaman@cisco.com>

   Editor:    Xufeng Liu
              <mailto:xufeng.liu.ietf@gmail.com>

   Editor:    Zhibo Hu
              <mailto:huzhibo@huawei.com>

   Editor:    Iftekhar Hussain
              <mailto:IHussain@infinera.com>

   Editor:    Himanshu Shah
              <mailto:hshah@ciena.com>

   Editor:    Daniel Voyer
              <mailto:daniel.voyer@bell.ca>

   Editor:    Hani Elmalky
              <mailto:helmalky@google.com>
```

Editor: Satoru Matsushima
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam
<mailto:ahabdel@cisro.com>

";

description

"This YANG module defines the essential elements for the management of Static application for Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).";

reference "RFC XXXX";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

revision 2020-07-13 {

description

"Alignment with SRv6 network programming rev16";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2019-10-30 {

description

"Extended model for EVPN behaviors";

reference

"RFC XXXX: YANG Data Model for SRv6";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

}

revision 2019-07-08 {

```
    description
      "Alignment with SRv6 network programming";
    reference
      "RFC XXXX: YANG Data Model for SRv6";
      // RFC Editor: replace XXXX with actual RFC number and remove
      // this note
  }

  revision 2018-10-22 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Data Model for SRv6";
      // RFC Editor: replace XXXX with actual RFC number and remove
      // this note
  }

/*
 * Config and State
 */

  grouping path-attrs-v6 {
    description
      "IPv6 Path properties";

    leaf interface {
      type if:interface-ref;
      description "The outgoing interface";
    }

    leaf next-hop {
      type inet:ipv6-address;
      description "The IP address of the next-hop";
    }

    leaf table {
      type srv6-types:table-id;
      description "The routing table associated with the next-hop";
    }

    uses srv6:path-attrs-cmn;
  }

  grouping path-attrs-v4 {
    description
      "IPv4 Path properties";
```

```
    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ipv4-address;
        description "The IP address of the next-hop";
    }

    leaf table {
        type srv6-types:table-id;
        description "The routing table associated with the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping path-attrs-mpls {
    description
        "MPLS Path properties";

    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ip-address;
        description "The IP address of the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping multi-paths-v6 {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }
        }
    }
}
```

```
        uses path-attrs-v6;
        container sid-list {
            description "SID-list associated with the path";
            uses srv6:path-out-sids;
        }
    }
}

grouping multi-paths-v4 {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            uses path-attrs-v4;
            container sid-list {
                description "SID-list associated with the path";
                uses srv6:path-out-sids;
            }
        }
    }
}

grouping multi-paths-mpls {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            uses path-attrs-mpls;
            container sid-list {
```

```
        description "SID-list associated with the path";
        uses srv6:path-out-sids;
    }
}
}

grouping multi-paths-v6-BUM {
    description
        "Multipath grouping for EVPN bridging BUM use case";

    container paths {
        description
            "List of outgoing paths for flooding";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            leaf l2-interface {
                type if:interface-ref;
                description "The outgoing L2 interface for flooding";
            }
        }
    }
}

grouping srv6-sid-config {
    description
        "Configuration parameters relating to SRv6 sid.";

    leaf function {
        type srv6-types:srv6-func-value;
        description
            "SRv6 function value.";
    }
    leaf end-behavior-type {
        type identityref {
            base srv6-types:srv6-endpoint-type;
        }
        mandatory true;
        description
            "Type of SRv6 end behavior.";
    }
}
```

```
container end {
  when "../end-behavior-type = 'End'" {
    description
      "This container is valid only when the user chooses End
      behavior (variant: no PSP, no USP).";
  }
  description
    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: no PSP, no USP)";
}

container end_psp {
  when "../end-behavior-type = 'End_PSP'" {
    description
      "This container is valid only when the user chooses End
      behavior (variant: PSP only).";
  }
  description
    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: PSP only)";
}

container end_usp {
  when "../end-behavior-type = 'End_USP'" {
    description
      "This container is valid only when the user chooses End
      behavior (variant: USP only).";
  }
  description
    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: USP only)";
}

container end_psp_usp {
  when "../end-behavior-type = 'End_PSP_USP'" {
    description
```

```
        "This container is valid only when the user chooses End
          behavior (variant: PSP/USP).";
    }
    description
      "The Endpoint function is the most basic function.
       FIB lookup on updated DA and forward accordingly
       to the matched entry.
       This is the SRv6 instantiation of a Prefix SID
       (variant: PSP/USP)";
  }

  container end_usd {
    when "../end-behavior-type = 'End_USD'" {
      description
        "This container is valid only when the user chooses End
          behavior (variant: USD only).";
    }
    description
      "The Endpoint function is the most basic function.
       FIB lookup on updated DA and forward accordingly
       to the matched entry.
       This is the SRv6 instantiation of a Prefix SID
       (variant: USD)";
  }

  container end_psp_usd {
    when "../end-behavior-type = 'End_PSP_USD'" {
      description
        "This container is valid only when the user chooses End
          behavior (variant: PSP/USD).";
    }
    description
      "The Endpoint function is the most basic function.
       FIB lookup on updated DA and forward accordingly
       to the matched entry.
       This is the SRv6 instantiation of a Prefix SID
       (variant: PSP/USD)";
  }

  container end_usp_usd {
    when "../end-behavior-type = 'End_USP_USD'" {
      description
        "This container is valid only when the user chooses End
          behavior (variant: USP/USD).";
    }
    description
```



```
    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: USP/USD)";

}

container end_psp_usp_usd {
  when "../end-behavior-type = 'End_PSP_USP_IUSD'" {
    description
      "This container is valid only when the user chooses End
      behavior (variant: PSP/USP/USD).";
  }
  description
    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: PSP/USP/USD)";
}

container end-t {
  when "../end-behavior-type = 'End.T'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: no PSP, no USP).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: no PSP,
    no USP).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

    // TODO presence "Mandatory child only if container is present";
    leaf lookup-table-ipv6 {
      type srv6-types:table-id;
      mandatory true;
      description
        "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp {
```

```
when "../end-behavior-type = 'End.T_PSP'" {
  description
    "This container is valid only when the user chooses
    End.T behavior (variant: PSP only).";
}
description
  "Endpoint with specific IPv6 table lookup (variant: PSP only).
  Lookup the next segment in IPv6 table T
  associated with the SID and forward via
  the matched table entry.

  The End.T is used for multi-table operation
  in the core.";

// TODO presence "Mandatory child only if container is present";

leaf lookup-table-ipv6 {
  type srv6-types:table-id;
  mandatory true;
  description
    "Table Id for lookup on updated DA (next segment)";
}

container end-t_usp {
  when "../end-behavior-type = 'End.T_USP'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USP only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USP only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}
```

```
container end-t_psp_usp {
  when "../end-behavior-type = 'End.T_PSP_USP'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USP/PSP).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USP/PSP).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

container end-t_usd {
  when "../end-behavior-type = 'End.T_USD'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USD only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}
```

```
container end-t_psp_usd {
  when "../end-behavior-type = 'End.T_PSP_USD'" {
    description
      "This container is valid only when the user chooses
       End.T behavior (variant: PSP/USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: PSP/USD
    only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

container end-t_usp_usd {
  when "../end-behavior-type = 'End.T_USP_USD'" {
    description
      "This container is valid only when the user chooses
       End.T behavior (variant: USP/USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant:
    USP/USD only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}
```

```
}

container end-t_psp_usp_usd {
  when "../end-behavior-type = 'End.T_PSP_USP_USD'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USP only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant:
    PSP/USP/USD only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
      type srv6-types:table-id;
      mandatory true;
      description
        "Table Id for lookup on updated DA (next segment)";
    }
  }

container end-x {
  when "../end-behavior-type = 'End.X'" {
    description
      "This container is valid only when the user chooses
      End.X behavior (variant: no USP/PSP)";
  }
  description
    "Endpoint with cross-connect to an array of
    layer-3 adjacencies (variant: no USP/PSP).
    Forward to layer-3 adjacency bound to the SID S.
    The End.X function is required to express any
    traffic-engineering policy.";

    leaf protected {
      type boolean;
      default false;
      description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
  }
}
```

```
container end-x_psp {
  when "../end-behavior-type = 'End.X_PSP'" {
    description
      "This container is valid only when the user chooses
       End.X behavior (variant: PSP only)";
  }
  description
    "Endpoint with cross-connect to an array of
     layer-3 adjacencies (variant: PSP only).
     Forward to layer-3 adjacency bound to the SID S.
     The End.X function is required to express any
     traffic-engineering policy.";

  leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
  }

  uses multi-paths-v6;
}

container end-x_usp {
  when "../end-behavior-type = 'End.X_USP'" {
    description
      "This container is valid only when the user chooses
       End.X behavior (variant: USP only)";
  }
  description
    "Endpoint with cross-connect to an array of
     layer-3 adjacencies (variant: USP only).
     Forward to layer-3 adjacency bound to the SID S.
     The End.X function is required to express any
     traffic-engineering policy.";

  leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
  }

  uses multi-paths-v6;
}

container end-x_psp_usp {
  when "../end-behavior-type = 'End.X_PSP_USP'" {
    description
      "This container is valid only when the user chooses
```

```
        End.X behavior (variant: PSP/USP)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_usd {
    when "../end-behavior-type = 'End.X_USD'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USD only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_psp_usd {
    when "../end-behavior-type = 'End.X_PSP_USD'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP/USD only)";
    }
    description
```

```
    "Endpoint with cross-connect to an array of
    layer-3 adjacencies (variant: PSP/USP).
    Forward to layer-3 adjacency bound to the SID S.
    The End.X function is required to express any
    traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_usp_usd {
    when "../end-behavior-type = 'End.X_USP_USD'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USP/USD only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_psp_usp_usd {
    when "../end-behavior-type = 'End.X_PSP_USP_USD'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP/USP/USD only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
```



```
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-b6-encaps {
    when "../end-behavior-type = 'End.B6.Encaps' or
        ../end-behavior-type = 'End.B6.Encaps.Red'" {
        description
            "This container is valid only when the user chooses
            End.B6.Encaps or End.B6.Encaps.Red behavior.";
    }
    description
        "Endpoint bound to an SRv6 Policy.
        Insert SRH based on the policy and forward the
        packet toward the first hop configured in the policy.
        This is the SRv6 instantiation of a Binding SID.
        This behavior also adds an outer IPv6 header";

    // TODO presence "Mandatory child only if container is present";

    leaf policy-name {
        type string;
        mandatory true;
        description "SRv6 policy name.";
    }
    leaf source-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "IPv6 source address for Encap.";
    }

    uses multi-paths-v6;
}

container end-bm {
    when "../end-behavior-type = 'End.BM'" {
        description
            "This container is valid only when the user chooses
            End.BM behavior.";
    }
}
```

```
description
  "Endpoint bound to an SR-MPLS Policy.
  push an MPLS label stack <L1, L2, L3> on the
  received packet and forward the according to
  Lable L1.
  This is an SRv6 instantiation of an SR-MPLS Binding SID.";

// TODO presence  "Mandatory child only if container is present";

leaf policy-name {
  type string;
  mandatory true;
  description "SRv6 policy name";
}
uses multi-paths-mpls;
}

container end-dx6 {
  when "../end-behavior-type = 'End.DX6'" {
    description
      "This container is valid only when the user chooses
      End.DX6 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
    an array of IPv6 adjacencies. Pop the (outer)
    IPv6 header and its extension headers and forward
    to layer-3 adjacency bound to the SID S.
    The End.DX6 used in the L3VPN use-case.";

  uses multi-paths-v6;
  // TODO: Backup path of type "Lookup in table"
}

container end-dx4 {
  when "../end-behavior-type = 'End.DX4'" {
    description
      "This container is valid only when the user chooses
      End.DX4 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
    an array of IPv4 adjacencies.
    Pop the (outer) IPv6 header and its extension
    header and forward to layer-3 adjacency bound
    to the SID S.
    This would be equivalent to the per-CE VPN
    label in MPLS.";
```

```
    uses multi-paths-v4;
    // TODO: Backup path of type "Lookup in table"
}
container end-dt6 {
    when "../end-behavior-type = 'End.DT6'" {
        description
            "This container is valid only when the user chooses
            End.DT6 behavior.";
    }
    description
        "Endpoint with decapsulation and specific IPv6 table
        lookup.
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv6 DA in IPv6
        table T and forward via the matched table entry.
        End.DT6 function is used in L3VPN use-case.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv6 table";
    }
}
container end-dt4 {
    when "../end-behavior-type = 'End.DT4'" {
        description
            "This container is valid only when the user chooses
            End.DT4 behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        IPv4 table lookup.
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv4 DA in IPv4
        table T and forward via the matched table entry.
        This would be equivalent to the per-VRF VPN label
        in MPLS.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv4 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv4 table";
    }
}
```

```
    }
  }
  container end-dt46 {
    when "../end-behavior-type = 'End.DT46'" {
      description
        "This container is valid only when the user chooses
        End.DT46 behavior.";
    }
    description
      "Endpoint with decapsulation and specific
      IP table lookup.
      Depending on the protocol type (IPv4 or IPv6)
      of the inner ip packet and the specific VRF name
      forward the packet.
      This would be equivalent to the per-VRF VPN
      label in MPLS.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv4 {
      type srv6-types:table-id;
      mandatory true;
      description "IPv4 table";
    }
    leaf lookup-table-ipv6 {
      type srv6-types:table-id;
      mandatory true;
      description "IPv6 table";
    }
  }
}

/* EVPN END behavior types */
container end-dx2 {
  when "../end-behavior-type = 'End.DX2'" {
    description
      "This container is valid only when the user chooses
      End.DX2 behavior.";
  }
  description
    "This is an Endpoint with decapsulation and Layer-2
    cross-connect to OIF.
    Pop the (outer) IPv6 header and its extension headers.
    Forward the resulting frame via OIF associated to the SID.
    The End.DX2 function is the L2VPN/EVPN VPWS use-case.";

  container path {
    description "Outgoing path";
    leaf l2-interface {
```

```
        type if:interface-ref;
        mandatory true;
        description "Outgoing L2 interface";
    }
}

container end-dx2v {
    when "../end-behavior-type = 'End.DX2V'" {
        description
            "This container is valid only when the user chooses
            End.DX2V behavior.";
    }
    description
        "Endpoint with decapsulation and specific VLAN
        L2 table lookup.
        Pop the (outer) IPv6 header and its extension headers.
        Lookup the exposed inner VLANs in L2 table T.
        Forward via the matched table entry.
        The End.DX2V is used for EVPN Flexible cross-connect
        use-cases";

    leaf lookup-table-vlan {
        type srv6-types:table-id;
        mandatory true;
        description
            "VLAN lookup table. There could be multiple
            vlan demux tables on the node, where a DX2V SID
            points to one vlan table";
    }
}

container end-dt2u {
    when "../end-behavior-type = 'End.DT2U'" {
        description
            "This container is valid only when the user chooses
            End.DT2U behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        unicast L2 MAC table lookup.
        Pop the (outer) IPv6 header and its extension headers.
        Learn the exposed inner MAC SA in L2 MAC table T.
        Lookup the exposed inner MAC DA in L2 MAC table T.
        Forward via the matched T entry else to all L2OIF in T.
        The End.DT2U is used for EVPN Bridging unicast use cases";

    leaf lookup-table-mac {
```

```
    type srv6-types:table-id;
    mandatory true;
    description "MAC L2 lookup table";
  }
}

container end-dt2m {
  when "../end-behavior-type = 'End.DT2M'" {
    description
      "This container is valid only when the user chooses
       End.DT2M behavior.";
  }
  description
    "Endpoint with decapsulation and specific flooding table.
     Pop the (outer) IPv6 header and its extension headers.
     Learn the exposed inner MAC SA in L2 MAC table T.
     Forward on all L2OIF (in the flooding table) excluding the one
     identified by Arg.FE2.
     The End.DT2M is used for EVPN Bridging BUM use case with
     ESI (Split Horizon) filtering capability.";

  leaf flooding-table {
    type srv6-types:table-id;
    mandatory true;
    description "L2 Flooding table (list of OIFs)";
  }

  uses multi-paths-v6-BUM;

  /* TODO - Support for argument Arg.FE2. It is an argument specific
   to EVPN ESI filtering and EVPN-ETREE used to exclude specific
   OIF (or set of OIFs) from flooding table. */
}

/* End of EVPN END behavior types */
}

grouping srv6-static-cfg {
  description
    "Grouping configuration and operation for SRv6 sid.";

  list sid {
    key "function";
    description "List of locally instantiated SIDs";

    uses srv6-sid-config;
  }
}
```

```
augment "/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator" {
  description
    "This augments locator leaf within SRv6.";

  container static {
    description "Static SRv6";

    /* Local SIDs */
    container local-sids {
      description
        "SRv6-static locally instantiated SIDs";

      uses srv6-static-cfg;
      /* no state for now; SID state accessible through base model */
    }
  }
}
} // module

<CODE ENDS>
```

Figure 7: ietf-srv6-static.yang

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

It goes without saying that this specification also inherits the security considerations captured in the SRv6 specification document [I-D.ietf-spring-srv6-network-programming].

7. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

| URI | Registrant | XML |
|--|------------|-----|
| urn:ietf:params:xml:ns:yang:ietf-srv6-types | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-srv6-base | The IESG | N/A |
| urn:ietf:params:xml:ns:yang:ietf-srv6-static | The IESG | N/A |

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

| Name | Namespace | Prefix | Reference |
|------------------|--|-------------|---------------|
| ietf-srv6-types | urn:ietf:params:xml:ns:yang:ietf-srv6-types | srv6-types | This document |
| ietf-srv6-base | urn:ietf:params:xml:ns:yang:ietf-srv6-base | srv6 | This document |
| ietf-srv6-static | urn:ietf:params:xml:ns:yang:ietf-srv6-static | srv6-static | This document |

-- RFC Editor: Replace "This document" with the document RFC number at time of publication, and remove this note.

8. Acknowledgments

The authors would like to acknowledge Darren Dukes, Les Ginsberg, Ahmed Bashandy, Rajesh Venkateswaran, and Mike Mallin for their review of some of the contents in this draft.

9. References

9.1. Normative References

- [I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Lindem, A., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-17 (work in progress), July 2020.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-16 (work in progress), June 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

9.2. Informative References

- [I-D.ietf-dmm-srv6-mobile-uplane]
Matsushima, S., Filsfils, C., Kohno, M., Camarillo, P., Voyer, D., and C. Perkins, "Segment Routing IPv6 for Mobile User Plane", draft-ietf-dmm-srv6-mobile-uplane-08 (work in progress), June 2020.
- [I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-ietf-spring-sr-service-programming-02 (work in progress), March 2020.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

Authors' Addresses

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Sonal Agarwal
Cisco Systems
Email: agarwaso@cisco.com

Xufeng Liu
Volta Networks
Email: xufeng.liu.ietf@gmail.com

Zhibo Hu
Huawei Technologies
Email: huzhibo@huawei.com

Iftekhar Hussain
Infinera Corporation
Email: IHussain@infinera.com

Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Hani Elmalky
Individual
Email: helmalky@google.com

Satoru Matsushima
SoftBank
Email: satoru.matsushima@g.softbank.co.jp

Katsuhiro Horiba
SoftBank
Email: katsuhiro.horiba@g.softbank.co.jp

Jaganbabu Rajamanickam
Cisco Systems
Email: jrajaman@cisco.com

Ahmed AbdelSalam
Cisco Systems
Email: ahabdels@cisco.com

SPRING Working Group
Internet-Draft
Intended status: Informational
Expires: May 5, 2020

T. Saad
V. Beeram
C. Barth
Juniper Networks, Inc.
November 02, 2019

Segment-Routing over Forwarding Adjacency Links
draft-saad-sr-fa-link-00

Abstract

Label Switched Paths (LSPs) set up in Multiprotocol Label Switching (MPLS) networks can be used to form Forwarding Adjacency (FA) links that carry traffic in those networks. An FA link can be assigned Traffic Engineering (TE) parameters that allow other LSR(s) to include it in their constrained path computation. FA link(s) can be also assigned Segment-Routing (SR) segments that enable the steering of traffic on to the associated FA link(s). The TE and SR attributes of an FA link can be advertised using known link state protocols. This document elaborates on the usage of FA link(s) and their attributes in SR enabled networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 5, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. Forwarding Adjacency Links | 3 |
| 3.1. Creation and Management | 3 |
| 3.2. Link Flooding | 4 |
| 3.3. Underlay LSP(s) | 4 |
| 3.4. State Changes | 4 |
| 3.5. TE Parameters | 5 |
| 3.6. Link Local and Remote Identifiers | 5 |
| 4. Segment-Routing over FA Links | 6 |
| 4.1. SR IGP Segments for FA | 6 |
| 4.1.1. Parallel Adjacencies | 7 |
| 4.2. SR BGP Segments for FA | 7 |
| 4.3. Applicability to Interdomain | 8 |
| 5. IANA Considerations | 8 |
| 6. Security Considerations | 8 |
| 7. Acknowledgement | 8 |
| 8. Normative References | 8 |
| Authors' Addresses | 10 |

1. Introduction

To improve scalability in Multi-Protocol Label Switching (MPLS) networks, it may be useful to create a hierarchy of LSPs as Forwarding Adjacencies (FA). The concept of FA link(s) and FA-LSP(s) was introduced in [RFC4206].

In Segment-Routing (SR), this is particularly useful for two main reasons.

First, it allows the stitching of sub-path(s) so as to realize an end-to-end SR path. Each sub-path can be represented by a FA link that is supported by one or more underlying LSP(s). The underlying LSP(s) that support an FA link can be setup using different technologies- including RSVP-TE, LDP, and SR. The sub-path(s), or FA link(s) in this case, can possibly interconnect multiple administrative domains, allowing each FA link within a domain to use a different technology to setup the underlying LSP(s).

Second, it allows shortening of a large SR Segment-List by compressing one or more slice(s) of the list into a corresponding FA link that each can be represented by a single segment- see Section 4. Effectively, it reduces the number of segments that an ingress router has to impose to realize an end-to-end path.

The FA links are treated as normal link(s) in the network and hence it can leverage existing link state protocol extensions to advertise properties associated with the FA link. For example, Traffic-Engineering (TE) link parameters and Segment-Routing (SR) segments parameters can be associated with the FA link and advertised throughout the network.

Once advertised in the network using a suitable link state protocol (such as OSPF, ISIS or BGP-LS), other LSR(s) in the network can use the FA TE link(s) as well as possibly other normal TE link(s) when performing path computation and/or when specifying the desired explicit path.

Though the concepts discussed in this document are specific to MPLS technology, these are also extensible to other dataplane technologies - e.g. SRv6.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Forwarding Adjacency Links

FA Link(s) can be created and supported by underlying FA LSPs. The FA link is of type point-to-point. FAs may be represented as either unnumbered or numbered links. The nodes connected by an FA link do not usually establish a routing adjacency over the FA link. When FAs are numbered with IPv4 addresses, the local and remote IPv4 addresses come out of a /31 that is allocated by the LSR that originates the FA-LSP. For unnumbered FA link(s), other provisions may exist to exchange link identifier(s) between the endpoints of the FA.

3.1. Creation and Management

In general, the creation/termination of an FA link and its FA-LSP is driven either via configuration on the LSR at the head-end of the adjacency, or dynamically using suitable North Bound Interface (NBI) protocol, e.g. Netconf, gRPC, PCEP, etc.

The following FA-LSP attributes may be configured, including: bandwidth and resource colors, and other constraints. The path taken by the FA-LSP may be either computed by the LSR at the head-end of the FA-LSP, or externally by a PCE and furnished to the headend.

The attributes of the FA link can be inherited from the underlying LSP(s) that induced its creation. In general, for dynamically provisioned FAs, a policy-based mechanism may be needed to associate link attributes to those of the FA-LSPs.

When the FA link is supported by bidirectional FA LSP(s), a pair of FA link(s) are advertised from each endpoint of the FA. These are usually referred to as symmetrical link(s).

3.2. Link Flooding

Multiple protocols exist that can exchange link state in the network. For example, when advertising TE link(s) and attribute(s) using OSPF and ISIS, the respective extensions are defined in [RFC3630] and [RFC5305]. Also, when exchanging such information in BGP, extensions for BGP link-state are defined in [RFC7752] and [RFC8571].

The same protocol encodings can be used to advertise an FA link. As a result, the FA TE link(s) and other normal TE link(s) will appear in the TE link state database of any LSR in the network.

3.3. Underlay LSP(s)

The LSR that hosts an FA link can setup the underlying LSP(s) using different technologies - e.g. RSVP-TE, LDP, and SR.

The FA link can be supported by one or more underlay LSP(s) that terminate on the same remote endpoint. The underlay path(s) can be setup using different signaling technologies, e.g. using RSVP-TE, LDP, SR, etc. When multiple LSP(s) support the same FA link, the attributes of the FA link can be derived from the aggregate properties of each of the underlying LSP(s).

3.4. State Changes

The state of an FA TE link reflects the state of the underlying LSP path that supports it. The TE link is assumed operational and is advertised as long as the underlying LSP path is valid. When all underlying LSP paths are invalidated, the FA TE link advertisement is withdrawn.

3.5. TE Parameters

The TE metrics and TE attributes are used by path computation algorithms to select the TE link(s) that a TE path traverses. When advertising an FA link in OSPF or ISIS, or BGP-LS, the following TE parameters are defined:

TE Path metrics: the FA link advertisement can include information about TE, IGP, and other performance metrics (e.g. delay, and loss). The FA link TE metrics, in this case, can be derived from the underlying path(s) that support the FA link by producing the path accumulative metrics. When multiple LSP(s) support the same FA link, then the higher accumulative metric amongst the LSP(s) is inherited by the FA link.

Resource Class/Color: An FA link can be assigned (e.g. via configuration) a specific set of admin-groups. Alternatively, in some cases, this can be derived from the underlying path affinity - for example, the underlying path strictly includes a specific admin-group.

SRLGs: An FA advertisement could contain the information about the Shared Risk Link Groups (SRLG) for the path taken by the FA LSP associated with that FA. This information may be used for path calculation by other LSRs. The information carried is the union of the SRLGs of the underlying TE links that make up the FA LSP path. It is possible that the underlying path information might change over time, via configuration updates or dynamic route modifications, resulting in the change of the union of SRLGs for the FA link. If multiple LSP(s) support the same FA link, then it is expected all LSP(s) have the same SRLG union - note, that the exact paths need not be the same.

It is worth noting, that topology changes in the network may affect the FA link underlying LSP path(s), and hence, can dynamically change the TE metrics and TE attributes of the FA links.

3.6. Link Local and Remote Identifiers

It is possible for the FA link to be numbered or unnumbered. [RFC4206] describes a procedure for identifying a numbered FA TE link using IPv4 addresses.

For unnumbered FA link(s), the assignment and handling of the local and remote link identifiers is specified in [RFC3477]. The LSR at each end of the unnumbered FA link assigns an identifier to that link. This identifier is a non-zero 32-bit number that is unique within the scope of the LSR that assigns it. There is no a priori

relationship between the identifiers assigned to a link by the LSRs at each end of that link.

The FA link is a unidirectional and point-to-point link. Hence, the combination of link local identifier and advertising node can uniquely identify the link in the TED. In some cases, however, it is desirable to associate the forward and reverse FA links in the TED. In this case, the combination of link local and remote identifier can identify the pair of forward and reverse FA link(s). The LSRs at the two end points of an unnumbered link can exchange with each other the identifiers they assign to the link. Exchanging the identifiers may be accomplished by configuration, or by means of protocol extensions. For example, when the FA link is established over RSVP-TE FA LSP(s), then RSVP extensions have been introduced to exchange the FA link identifier in [RFC3477]. Other protocol extensions pertaining to specific link state protocols, and LSP setup technologies will be discussed in a separate document.

If the link remote identifier is unknown, the value advertised is set to 0 [RFC5307].

4. Segment-Routing over FA Links

The Segment Routing (SR) architecture [RFC4206] describes that an IGP adjacency can be formed over a FA link – in which the remote node of an IGP adjacency is a non-adjacent IGP neighbor.

In Segment-Routing (SR), the adjacency that is established over a link can be assigned an SR Segment [RFC8402]. For example, the Adj-SID allows to strictly steer traffic on to the specific adjacency that is associated with the Adj-SID.

4.1. SR IGP Segments for FA

Extensions have been defined to ISIS [I-D.ietf-isis-segment-routing-extensions] and OSPF [I-D.ietf-ospf-segment-routing-extensions] in order to advertise the the Adjacency-SID associated with a specific IGP adjacency. The same extensions apply to adjacencies over FA link. A node can bind an Adj-SID to an FA data-link. The Adj-SID dictates the forwarding of packets through the specific FA link or FA link(s) identified by the Adj-SID, regardless of its IGP/SPF cost.

When the FA link Adj-SID is supported by a single underlying LSP that is associated with a binding label or SID, the same binding label can be used for the FA link Adj-SID. For example, if the FA link is supported by an SR Policy that is assigned a Binding SID B, the Adj-SID of the FA link can be assigned the same Binding SID B.

When the FA link Adj-SID is supported by multiple underlying LSP(s) or SR Policies – each having its own Binding label or SID, an independent FA link Adj-SID is allocated and bound to the multiple underlying LSP(s).

4.1.1. Parallel Adjacencies

Adj-SIDs can also be used in order to represent a set of parallel FA link(s) between two endpoints.

When parallel FA links are associated with the same Adj-SID, a "weight" factor can be assigned to each link and advertised with the Adj-SID advertised with each FA link. The weight informs the ingress (or an SDN/orchestration system) about the load-balancing factor over the parallel adjacencies.

4.2. SR BGP Segments for FA

BGP segments are allocated and distributed by BGP. The SR architecture [RFC8402] defines three types of BGP segments for Egress Peer Engineering (EPE): PeerNode SID, PeerAdj SID, and PeerSet SID.

The applicability of each of the three types to FA links is discussed below:

o PeerNode SID: a BGP PeerNode segment/SID is a local segment. At the BGP node advertising, the forwarding semantics are:

- * SR operation: NEXT.
- * Next-Hop: forward over any FA link associated with the segment that terminates on remote endpoint.

o PeerAdj SID: a BGP PeerAdj segment/SID is a local segment. At the BGP node advertising it, the forwarding semantics are:

- * SR operation: NEXT.
- * Next-Hop: forward over the specific FA link to the remote endpoint to which the segment is related.

o PeerSet SID: a BGP PeerSet segment/SID is a local segment. At the BGP node advertising it, the semantics are:

- * SR operation: NEXT.
- * Next-Hop: load-balance across any of the FA links to any remote endpoint in the related set. The group definition is a policy set by the operator.

4.3. Applicability to Interdomain

In order to determine the potential to establish a TE path through a series of interconnected domains or multi-domain network, it is necessary to have available a certain amount of TE information about each network domain. This need not be the full set of TE information available within each network but does need to express the potential of providing such TE connectivity.

Topology abstraction is described in [RFC7926]. Abstraction allows applying a policy to the available TE information within a domain so to produce selective information that represents the potential ability to connect across the domain. Thus, abstraction does not necessarily offer all possible connectivity options, but presents a general view of potential connectivity according to the policies that determine how the domain's administrator wants to allow the domain resources to be used.

Hence, the domain may be constructed as a mesh of border node to border node TE FA links. When computing a path for an LSP that crosses the domain, a computation point can see which domain entry points can be connected to which others, and with what TE attributes.

5. IANA Considerations

TBD.

6. Security Considerations

TBD.

7. Acknowledgement

TBD.

8. Normative References

- [I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-25 (work in progress), May 2019.

- [I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,
Shakir, R., Henderickx, W., and J. Tantsura, "OSPF
Extensions for Segment Routing", draft-ietf-ospf-segment-
routing-extensions-27 (work in progress), December 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3477] Kompella, K. and Y. Rekhter, "Signalling Unnumbered Links
in Resource ReSerVation Protocol - Traffic Engineering
(RSVP-TE)", RFC 3477, DOI 10.17487/RFC3477, January 2003,
<<https://www.rfc-editor.org/info/rfc3477>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering
(TE) Extensions to OSPF Version 2", RFC 3630,
DOI 10.17487/RFC3630, September 2003,
<<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP)
Hierarchy with Generalized Multi-Protocol Label Switching
(GMPLS) Traffic Engineering (TE)", RFC 4206,
DOI 10.17487/RFC4206, October 2005,
<<https://www.rfc-editor.org/info/rfc4206>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic
Engineering", RFC 5305, DOI 10.17487/RFC5305, October
2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5307] Kompella, K., Ed. and Y. Rekhter, Ed., "IS-IS Extensions
in Support of Generalized Multi-Protocol Label Switching
(GMPLS)", RFC 5307, DOI 10.17487/RFC5307, October 2008,
<<https://www.rfc-editor.org/info/rfc5307>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and
S. Ray, "North-Bound Distribution of Link-State and
Traffic Engineering (TE) Information Using BGP", RFC 7752,
DOI 10.17487/RFC7752, March 2016,
<<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7926] Farrel, A., Ed., Drake, J., Bitar, N., Swallow, G.,
Ceccarelli, D., and X. Zhang, "Problem Statement and
Architecture for Information Exchange between
Interconnected Traffic-Engineered Networks", BCP 206,
RFC 7926, DOI 10.17487/RFC7926, July 2016,
<<https://www.rfc-editor.org/info/rfc7926>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.

Authors' Addresses

Tarek Saad
Juniper Networks, Inc.

Email: tsaad@juniper.net

Vishnu Pavan Beeram
Juniper Networks, Inc.

Email: vbeeram@juniper.net

Colby Barth
Juniper Networks, Inc.

Email: cbarth@juniper.net

SPRING Working Group
Internet-Draft
Intended status: Informational
Expires: August 19, 2021

T. Saad
V. Beeram
C. Barth
Juniper Networks, Inc.
S. Sivabalan
Ciena Corporation.
February 15, 2021

Segment-Routing over Forwarding Adjacency Links
draft-saad-sr-fa-link-03

Abstract

Label Switched Paths (LSPs) set up in Multiprotocol Label Switching (MPLS) networks can be used to form Forwarding Adjacency (FA) links that carry traffic in those networks. An FA link can be assigned Traffic Engineering (TE) parameters that allow other LSR(s) to include it in their constrained path computation. FA link(s) can be also assigned Segment-Routing (SR) segments that enable the steering of traffic on to the associated FA link(s). The TE and SR attributes of an FA link can be advertised using known protocols that carry link state information. This document elaborates on the usage of FA link(s) and their attributes in SR enabled networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 19, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. Forwarding Adjacency Links | 3 |
| 3.1. Creation and Management | 4 |
| 3.2. Link Flooding | 4 |
| 3.3. Underlay LSP(s) | 5 |
| 3.4. State Changes | 5 |
| 3.5. TE Parameters | 5 |
| 3.6. Link Local and Remote Identifiers | 6 |
| 4. Segment-Routing over FA Links | 6 |
| 4.1. SR IGP Segments for FA | 7 |
| 4.1.1. Parallel Adjacencies | 7 |
| 4.2. SR BGP Segments for FA | 7 |
| 4.3. Applicability to Interdomain | 8 |
| 5. IANA Considerations | 9 |
| 6. Security Considerations | 9 |
| 7. Acknowledgement | 9 |
| 8. Normative References | 9 |
| Authors' Addresses | 10 |

1. Introduction

To improve scalability in Multi-Protocol Label Switching (MPLS) networks, it may be useful to create a hierarchy of LSPs as Forwarding Adjacencies (FA). The concept of FA link(s) and FA-LSP(s) was introduced in [RFC4206].

In Segment-Routing (SR), this is particularly useful for two main reasons.

First, it allows the stitching of sub-path(s) so as to realize an end-to-end SR path. Each sub-path can be represented by a FA link that is supported by one or more underlying LSP(s). The underlying LSP(s) that support an FA link can be setup using different technologies- including RSVP-TE, LDP, and SR. The sub-path(s), or FA link(s) in this case, can possibly interconnect multiple

administrative domains, allowing each FA link within a domain to use a different technology to setup the underlying LSP(s).

Second, it allows shortening of a large SR Segment-List by compressing one or more slice(s) of the list into a corresponding FA TE link that each can be represented by a single segment- see Section 4. Effectively, it reduces the number of segments that an ingress router has to impose to realize an end-to-end path.

The FA links are treated as normal link(s) in the network and hence it can leverage existing link state protocol extensions to advertise properties associated with the FA link. For example, Traffic-Engineering (TE) link parameters and Segment-Routing (SR) segments parameters can be associated with the FA link and advertised throughout the network.

Once advertised in the network using a suitable protocols that support carrying link state information, such as OSPF, ISIS or BGP Link State (LS)), other LSR(s) in the network can use the FA TE link(s) as well as possibly other normal TE link(s) when performing path computation and/or when specifying the desired explicit path.

Though the concepts discussed in this document are specific to MPLS technology, these are also extensible to other dataplane technologies - e.g. SRv6.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Forwarding Adjacency Links

FA Link(s) can be created and supported by underlying FA LSPs. The FA link is of type point-to-point. FA links may be represented as either unnumbered or numbered. The nodes connected by an FA link do not usually establish a routing adjacency over the FA link. When FA links are numbered with IPv4 addresses, the local and remote IPv4 addresses can come out of a /31 that is allocated by the LSR that originates the FA-LSP. For unnumbered FA link(s), other provisions may exist to exchange link identifier(s) between the endpoints of the FA.

3.1. Creation and Management

In general, the creation/termination of an FA link and its FA-LSP is driven either via configuration on the LSR at the head-end of the adjacency, or dynamically using suitable North Bound Interface (NBI) protocol, e.g. Netconf, gRPC, PCEP, etc.

The following FA-LSP attributes may be configured, including: bandwidth and resource colors, and other constraints. The path taken by the FA-LSP may be either computed by the LSR at the head-end of the FA-LSP, or externally by a PCE and furnished to the headend.

The attributes of the FA link can be inherited from the underlying LSP(s) that induced its creation. In general, for dynamically provisioned FAs, a policy-based mechanism may be needed to associate link attributes to those of the FA-LSPs.

When the FA link is supported by bidirectional FA LSP(s), a pair of FA link(s) are advertised from each endpoint of the FA. These are usually referred to as symmetrical link(s).

3.2. Link Flooding

Multiple protocols exist that can exchange link state information in the network. For example, when advertising TE link(s) and their attribute(s) using OSPF and ISIS protocols, the respective extensions are defined in [RFC3630] and [RFC5305]. Also, when exchanging such information in BGP protocol, extensions for BGP link state are defined in [RFC7752] and [RFC8571]. The same protocol encodings can be used to advertise FA(s) as TE link(s). As a result, the FA TE link(s) and other normal TE link(s) will appear in the TE link state database of any LSR in the network, and can be used for computing end-to-end TE path(s).

When IGP protocols are used to advertise link state information about FA links, the FA link(s) can appear in both the TE topology, as well as the IGP topology. The use of FA link in the IGP topology may result in undesirable routing loops. A router SHOULD leverage existing mechanisms to exclude the FA link from the IGP Shortest Path First (SPF) computations, and to restrict its use within the TE topology for traffic engineered paths computation.

For example, when using ISIS to carry FA link state information, [RFC5305] section 3 describes a way to restrict the link to the TE topology by setting the IGP link metric to maximum ($2^{24} - 1$). Alternatively, when using OSPF, the FA link(s) can be advertised using TE Opaque LSA(s) only, and hence, strictly show up in the TE topology as described in [RFC3630].

3.3. Underlay LSP(s)

The LSR that hosts an FA link can setup the underlying LSP(s) using different technologies - e.g. RSVP-TE, LDP, and SR.

The FA link can be supported by one or more underlay LSP(s) that terminate on the same remote endpoint. The underlay path(s) can be setup using different signaling technologies, e.g. using RSVP-TE, LDP, SR, etc. When multiple LSP(s) support the same FA link, the attributes of the FA link can be derived from the aggregate properties of each of the underlying LSP(s).

3.4. State Changes

The state of an FA TE link reflects the state of the underlying LSP path that supports it. The TE link is assumed operational and is advertised as long as the underlying LSP path is valid. When all underlying LSP paths are invalidated, the FA TE link advertisement is withdrawn.

3.5. TE Parameters

The TE metrics and TE attributes are used by path computation algorithms to select the TE link(s) that a TE path traverses. When advertising an FA link in OSPF or ISIS, or BGP-LS, the following TE parameters are defined:

TE Path metrics: the FA link advertisement can include information about TE, IGP, and other performance metrics (e.g. delay, and loss). The FA link TE metrics, in this case, can be derived from the underlying path(s) that support the FA link by producing the path accumulative metrics. When multiple LSP(s) support the same FA link, then the higher accumulative metric amongst the LSP(s) is inherited by the FA link.

Resource Class/Color: An FA link can be assigned (e.g. via configuration) a specific set of admin-groups. Alternatively, in some cases, this can be derived from the underlying path affinity - for example, the underlying path strictly includes a specific admin-group.

SRLGs: An FA advertisement could contain the information about the Shared Risk Link Groups (SRLG) for the path taken by the FA LSP associated with that FA. This information may be used for path calculation by other LSRs. The information carried is the union of the SRLGs of the underlying TE links that make up the FA LSP path. It is possible that the underlying path information might change over time, via configuration updates or dynamic route

modifications, resulting in the change of the union of SRLGs for the FA link. If multiple LSP(s) support the same FA link, then it is expected all LSP(s) have the same SRLG union - note, that the exact paths need not be the same.

It is worth noting, that topology changes in the network may affect the FA link underlying LSP path(s), and hence, can dynamically change the TE metrics and TE attributes of the FA links.

3.6. Link Local and Remote Identifiers

It is possible for the FA link to be numbered or unnumbered. [RFC4206] describes a procedure for identifying a numbered FA TE link using IPv4 addresses.

For unnumbered FA link(s), the assignment and handling of the local and remote link identifiers is specified in [RFC3477]. The LSR at each end of the unnumbered FA link assigns an identifier to that link. This identifier is a non-zero 32-bit number that is unique within the scope of the LSR that assigns it. There is no a priori relationship between the identifiers assigned to a link by the LSRs at each end of that link.

The FA link is a unidirectional and point-to-point link. Hence, the combination of link local identifier and advertising node can uniquely identify the link in the TED. In some cases, however, it is desirable to associate the forward and reverse FA links in the TED. In this case, the combination of link local and remote identifier can identify the pair of forward and reverse FA link(s). The LSRs at the two end points of an unnumbered link can exchange with each other the identifiers they assign to the link. Exchanging the identifiers may be accomplished by configuration, or by means of protocol extensions. For example, when the FA link is established over RSVP-TE FA LSP(s), then RSVP extensions have been introduced to exchange the FA link identifier in [RFC3477]. Other protocol extensions pertaining to specific link state protocols, and LSP setup technologies will be discussed in a separate document.

If the link remote identifier is unknown, the value advertised is set to 0 [RFC5307].

4. Segment-Routing over FA Links

The Segment Routing (SR) architecture [RFC4206] describes that an IGP adjacency can be formed over a FA link - in which the remote node of an IGP adjacency is a non-adjacent IGP neighbor.

In Segment-Routing (SR), the adjacency that is established over a link can be assigned an SR Segment [RFC8402]. For example, the Adj-SID allows to strictly steer traffic on to the specific adjacency that is associated with the Adj-SID.

4.1. SR IGP Segments for FA

Extensions have been defined to ISIS [RFC8667] and OSPF [RFC8665] in order to advertise the the Adjacency-SID associated with a specific IGP adjacency. The same extensions apply to adjacencies over FA link. A node can bind an Adj-SID to an FA data-link. The Adj-SID dictates the forwarding of packets through the specific FA link or FA link(s) identified by the Adj-SID, regardless of its IGP/SPF cost.

When the FA link Adj-SID is supported by a single underlying LSP that is associated with a binding label or SID, the same binding label can be used for the FA link Adj-SID. For example, if the FA link is supported by an SR Policy that is assigned a Binding SID B, the Adj-SID of the FA link can be assigned the same Binding SID B.

When the FA link Adj-SID is supported by multiple underlying LSP(s) or SR Policies - each having its own Binding label or SID, an independent FA link Adj-SID is allocated and bound to the multiple underlying LSP(s).

4.1.1. Parallel Adjacencies

Adj-SIDs can also be used in order to represent a set of parallel FA link(s) between two endpoints.

When parallel FA links are associated with the same Adj-SID, a "weight" factor can be assigned to each link and advertised with the Adj-SID advertised with each FA link. The weight informs the ingress (or an SDN/orchestration system) about the load-balancing factor over the parallel adjacencies.

4.2. SR BGP Segments for FA

BGP segments are allocated and distributed by BGP. The SR architecture [RFC8402] defines three types of BGP segments for Egress Peer Engineering (EPE): PeerNode SID, PeerAdj SID, and PeerSet SID.

The applicability of each of the three types to FA links is discussed below:

- o PeerNode SID: a BGP PeerNode segment/SID is a local segment. At the BGP node advertising, the forwarding semantics are:

- * SR operation: NEXT.
 - * Next-Hop: forward over any FA link associated with the segment that terminates on remote endpoint.
- o PeerAdj SID: a BGP PeerAdj segment/SID is a local segment. At the BGP node advertising it, the forwarding semantics are:

- * SR operation: NEXT.
 - * Next-Hop: forward over the specific FA link to the remote endpoint to which the segment is related.
- o PeerSet SID: a BGP PeerSet segment/SID is a local segment. At the BGP node advertising it, the semantics are:

- * SR operation: NEXT.
- * Next-Hop: load-balance across any of the FA links to any remote endpoint in the related set. The group definition is a policy set by the operator.

4.3. Applicability to Interdomain

In order to determine the potential to establish a TE path through a series of interconnected domains or multi-domain network, it is necessary to have available a certain amount of TE information about each network domain. This need not be the full set of TE information available within each network but does need to express the potential of providing such TE connectivity.

Topology abstraction is described in [RFC7926]. Abstraction allows applying a policy to the available TE information within a domain so to produce selective information that represents the potential ability to connect across the domain. Thus, abstraction does not necessarily offer all possible connectivity options, but presents a general view of potential connectivity according to the policies that determine how the domain's administrator wants to allow the domain resources to be used.

Hence, the domain may be constructed as a mesh of border node to border node TE FA links. When computing a path for an LSP that crosses the domain, a computation point can see which domain entry points can be connected to which others, and with what TE attributes.

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

TBD.

7. Acknowledgement

The authors would like to thank Peter Psenak for reviewing and providing valuable feedback on this document.

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3477] Kompella, K. and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, DOI 10.17487/RFC3477, January 2003, <<https://www.rfc-editor.org/info/rfc3477>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, DOI 10.17487/RFC4206, October 2005, <<https://www.rfc-editor.org/info/rfc4206>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5307] Kompella, K., Ed. and Y. Rekhter, Ed., "IS-IS Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 5307, DOI 10.17487/RFC5307, October 2008, <<https://www.rfc-editor.org/info/rfc5307>>.

- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7926] Farrel, A., Ed., Drake, J., Bitar, N., Swallow, G., Ceccarelli, D., and X. Zhang, "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", BCP 206, RFC 7926, DOI 10.17487/RFC7926, July 2016, <<https://www.rfc-editor.org/info/rfc7926>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.
- [RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <<https://www.rfc-editor.org/info/rfc8665>>.
- [RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <<https://www.rfc-editor.org/info/rfc8667>>.

Authors' Addresses

Tarek Saad
Juniper Networks, Inc.

Email: tsaad@juniper.net

Vishnu Pavan Beeram
Juniper Networks, Inc.

Email: vbeeram@juniper.net

Colby Barth
Juniper Networks, Inc.

Email: cbarth@juniper.net

Siva Sivabalan
Ciena Corporation.

Email: ssivabal@ciena.com

SPRING
Internet-Draft
Intended status: Informational
Expires: May 2, 2020

R. Nakamura, Ed.
The University of Tokyo
Y. Ueno
NTT Communications Corporation
T. Kamata
Cisco Systems, Inc.
October 30, 2019

An Experiment of SRv6 Service Chaining at Interop Tokyo 2019 ShowNet
draft-upa-srv6-service-chaining-exp-00

Abstract

This document reports lessons learned from an experimental deployment of service chaining with Segment Routing over the IPv6 data plane (SRv6) at an event network. The service chaining part of the network was comprised of four SRv6-capable nodes (three products from different vendors), five SRv6 proxy nodes (two products from different vendors and three open source software), and six services. This network was deployed at Interop Tokyo 2019, and it successfully provided network connectivity and services to all the exhibitors and visitors on the event.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 2, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. SRv6 service chaining at Interop Tokyo 2019 ShowNet | 3 |
| 4. Lessons Learned | 5 |
| 4.1. Transparency of SRv6 header | 5 |
| 4.2. Services that cannot co-exist with End.AM | 6 |
| 4.3. Service liveness detection and conditional advertisement of service segments | 6 |
| 4.4. TTL Decrement on SRv6 Proxies | 6 |
| 4.5. Control Plane Capabilities | 7 |
| 4.6. Match Condition for Applying SRv6 Functions | 7 |
| 5. IANA Considerations | 8 |
| 6. Security Considerations | 8 |
| 7. Contributors | 8 |
| 8. Acknowledgements | 8 |
| 9. Normative References | 8 |
| Authors' Addresses | 10 |

1. Introduction

Standardizing functionalities for SRv6 service programming is still an ongoing agenda. Meanwhile, some fundamental parts have begun to be implemented: basic transit behaviors, endpoint functions at ingress and egress nodes [I-D.filsfils-spring-srv6-network-programming], and SR proxies for SR-unaware services [I-D.ietf-spring-sr-service-programming]. Trying out such running codes and devices would clarify statuses of recent implementations and provide feedback to current and future standardization processes.

To clarify the current status, we conducted an experiment of SRv6 service chaining at Interop Tokyo. Interop Tokyo is a large exhibition of networking technologies, and ShowNet is the event network built at Interop Tokyo while demonstrating new technologies and conducting interoperability tests. In 2019, we deployed SRv6 service chaining with the latest implementations at ShowNet and provided Internet connectivity for over 200 exhibitors and over 155,000 visitors. Through the experiment, we made several

observations from both implementation and specification perspectives: transparency of SRv6 header (SRH), services that cannot co-exist with the original masquerading proxy, how to integrate service liveness into advertisement of service segments, behaviors of TTL decrement on the masquerading proxy, necessity of control planes, and behavior of the longest prefix match and SRv6 functions. This document reports and discusses these lessons learned from the experiment of SRv6 service chaining.

2. Terminology

This document leverages the terminology proposed in [RFC8402], [I-D.ietf-spring-segment-routing-policy], and [I-D.ietf-spring-sr-service-programming].

3. SRv6 service chaining at Interop Tokyo 2019 ShowNet

The experiment was conducted on Interop Tokyo, from June 12 to June 14, 2019. This section describes the overview of SRv6 service chaining at ShowNet 2019.

The devices contributed to the experiment are listed below:

| FUNCTION | NAME | CONTRIBUTOR |
|-------------------|-----------------------------------|--------------------|
| T.Insert | FX201 | Furukawa Electric |
| T.Encaps | FX201 | Furukawa Electric |
| End.DT4 | NCS55A1 | Cisco Systems |
| | NE40E-F1A | Huawei |
| End | FX201 | Furukawa Electric |
| End.AM | FX201 | Furukawa Electric |
| | Kamuee | NTT Communications |
| | VPP | FD.io |
| End.AN | TM VNFS | Trend Micro |
| Variant of End.AD | Two open source software on Linux | |

The variant of End.AD was the SRv6 tagging proxy designed for IPv4 traffic encapsulated in SRv6 and implemented for ShowNet. The detail is described in [I-D.eden-srv6-tagging-proxy]. In addition to the SRv6-capable devices, we deployed six SR-unaware services into the service chaining. These services were applied to user traffic in accordance with service menus.

| SERVICE | NAME | CONTRIBUTOR |
|----------|---------------------|--------------------|
| Security | FortiGate 3601E | Fortinet |
| | Lastline Defender | Lastline |
| | PA-5280 | Palo Alto Networks |
| | Thunder 3230S CFW | A10 Networks |
| | TBA | |
| CGN | Thunder 7440-11 CFW | A10 Networks |

Note that the detailed security services were different depending on each device (e.g., DPI, URL filtering, etc).

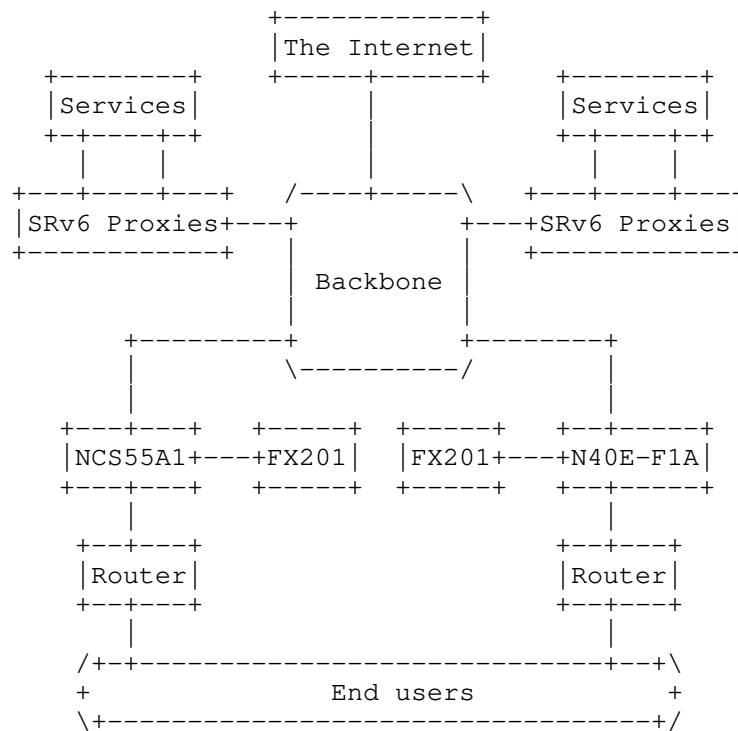


Figure 1: Overview of SRv6 Service Chaining at ShowNet

Figure 1 illustrates an overview of the experimental topology. End users, i.e., exhibitors and visitors, were accommodated by two non-SR-capable routers. The two FX201 had default routes for users, and the FX201 applied T.Insert and T.Encaps to received IPv6 and IPv4 packets from the users, respectively. The packets with the SRH were delivered to the SRv6 proxies by the active service segments. The SRv6 proxies connected to the backbone received the packets and

performed the proxy behaviors to take the packets through the services. Lastly, Segment List[0] representing endpoint functions took the packets to FX201 again for End and popping the SRH, or NCS55A1 or NE40E-F1A for End.DT4.

From the Internet to users, packets followed a similar path: FX201 inserted SRH to the packets, services were applied to the packets in the reverse order, and FX201 removed the SRH from IPv6 packets, or NCS55A1 or NE40E-F1A decapsulated IPv4 packets in the SRH and outer IPv6 headers.

The reason why we chose T.Insert instead of T.Encaps for IPv6 packets is to use the masquerading proxy (End.AM). SR proxies excluding End.AM require a proxy instance or an interface for receiving traffic returning from the service (IFACE-IN) for each SR service policy. In the ShowNet, there were over 200 individual users (exhibitors); therefore, there was the possibility that we needed to prepare over 200 proxy instances or interfaces for each service. It was possible, however, not reasonable for the temporal network for the three-days event. Therefore, we used T.Insert and End.AM for IPv6 packets to multiplex service chains on a proxy instance for a service.

End.AM is applicable to only SRv6 insertion; thus, IPv4 traffic still has the same issue. To address this issue, we designed the SRv6 tagging proxy, which is a new variant of End.AD, for IPv4 packets encapsulated in SRv6 [I-D.eden-srv6-tagging-proxy]. An XDP-based implementation was used for delivering all user traffic, and a Linux kernel-based implementation was also tested at ShowNet.

Although we faced some challenges described in the next section, this SRv6 service chaining finally fulfilled the role. It delivered all user traffic and applied the services during the period of Interop Tokyo 2019.

4. Lessons Learned

This section shares lessons learned from the experimental deployment.

4.1. Transparency of SRv6 header

First, we report that all the services contributed to ShowNet transparently delivered IPv6 packets under the End.AM proxies, although SRH is a new IPv6 extension header. There were no devices that dropped packets due to the unrecognized extension header.

4.2. Services that cannot co-exist with End.AM

When we designed the ShowNet, we intended to deploy a captive portal at service chaining. However, we could not accomplish that. The masquerading proxy assumes that the service under the proxy can only inspect, drop, or perform limited changes to the packets as noted in [I-D.ietf-spring-sr-service-programming]. Besides, it assumes that the packets returning from IFACE-IN have masqueraded SRH. This means that packets originated by SR-unaware services do not have SRH; therefore, the packets cannot be de-masqueraded. A captive portal is one of the examples that originate packets. These SR-unaware services cannot be integrated with the original masquerading proxy.

Instead, the variant 2 of masquerading proxy (Caching) defined in Section 6.4.3 of [I-D.ietf-spring-sr-service-programming] would be capable of handling such services.

4.3. Service liveness detection and conditional advertisement of service segments

Advertising service segments should be stopped when corresponding services are down to achieve redundancy of services in SRv6 service chaining. It is also expected that SR proxies are capable of stopping service segment advertisements. Meanwhile, the proxies contributed to ShowNet had not implemented such functionality yet because they were focusing on data plane functionalities at that time.

Link downs do not always represent service downs; services might be physical appliances behind switches or virtual machines on hypervisors. To detect failures on the services, we suggest that SR proxies should have some probing mechanisms for determining the statuses of services under the proxies and integrate the statuses with the advertisement process of the service segments. For example, Bidirectional Forwarding Detection (BFD) [RFC8562] between IFACE-IN and IFACE-OUT would determine the liveness of the services, and the liveness property could be integrated into triggers for advertising corresponding service segments.

4.4. TTL Decrement on SRv6 Proxies

We confirmed that there were variations on TTL decrement behaviors of the End.AM proxies. An implementation decreases TTL of outer IPv6 headers when sending packets to IFACE-OUT, and another implementation does not decrease TTL on IFACE-OUT. TTL decrement also occurs for forwarding packets from IFACE-IN to the next segments. As a result, TTL values of packets varied depending on the proxy implementations that the packets passed through. The variation of TTL decrement is

not a significant matter for just forwarding traffic; however, it would complicate Operation, Administration, and Maintenance (OAM) because traceroute results would also vary depending on implementations. Forwarding packets to IFACE-OUT is also layer-3 processing based on IPv6 headers and SRH; therefore, we suggest that TTL should be decreased on IFACE-OUT.

4.5. Control Plane Capabilities

The SR node implementations in ShowNet were not capable of control planes; therefore, we configured all the SR policies in ShowNet manually. Despite manually configuring SR nodes for service chaining is possible, it is hard to operate in realistic environments.

Advertising SR policies via BGP

[I-D.ietf-idr-segment-routing-te-policy] is also an essential capability for service chaining that is a work in progress

[I-D.dawra-idr-bgp-ls-sr-service-segments].

4.6. Match Condition for Applying SRv6 Functions

The SRv6 node implementations contributed to ShowNet applied SRv6 functions by the longest prefix match: SRv6 functions are represented by pairs of destination prefixes for segments and functions. Packets from users to the Internet would have arbitrary destinations; therefore, the transit behaviors must be associated with default routes. On the other hand, SR service policies inserted to packets vary depending on users and their service menus. To apply different SR service policies by the transit behaviors with default routes, we used VRFs on FX201 that performed T.Insert and T.Encaps. An SR service policy was associated with a transit behavior for a default route on a VRF. The user networks were attached to VRFs corresponding to their service menus. By this technique, we accomplished per-user service chains at the transit nodes.

Per-VRF transit behavior is a practical candidate for SRv6 service chaining. On the other hand, users and their traffic can be distinguished by source addresses of packets. Thus, applying SRv6 functions in accordance with source addresses or other fields in packets can also be a candidate implementation. For example, leveraging SRv6 functions as actions of BGP Flowspec [RFC5575] is a possible way for inserting SR policies into packets flexibly. This is a different adaptation of BGP Flowspec to SRv6 in addition to [I-D.ietf0-idr-srv6-flowspec-path-redirect].

5. IANA Considerations

This document has no IANA implications.

6. Security Considerations

The security requirements and mechanisms described in [RFC8402], [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming] also apply to this document.

This document does not introduce any new security vulnerabilities.

7. Contributors

J. Cao (Huawei), T. Fujiwara (Furukawa Network Solution), M. Udaka (Furukawa Network Solution), Y. Oga (Furukawa Network Solution), Y. Ohara (NTT Communications), H. Shirokura (NTT Communications), Y. Yamada (Keysight Technologies), K. Noda (Keysight Technologies), L. Zhou (Spirent), T. Kawada (TOYO Corporation), T. Matsuba (TOYO Corporation), Y. Matsubayashi (Trend Micro), and H. Nishikawa (Trend Micro) substantially contributed to the content of this document.

8. Acknowledgements

The authors would like to thank all the members and contributors of Interop Tokyo 2019 ShowNet. The authors are thankful to Francois Clad for his comments.

9. Normative References

[I-D.dawra-idr-bgp-ls-sr-service-segments]
Dawra, G., Filsfils, C., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., Elmalky, H., Xu, X., Clad, F., and K. Talaulikar, "BGP-LS Advertisement of Segment Routing Service Segments", draft-dawra-idr-bgp-ls-sr-service-segments-02 (work in progress), July 2019.

[I-D.eden-srv6-tagging-proxy]
Ueno, Y., Nakamura, R., and T. Kamata, "SRv6 Tagging proxy", draft-eden-srv6-tagging-proxy-00 (work in progress), October 2019.

- [I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.
- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment
Routing Header (SRH)", draft-ietf-6man-segment-routing-
header-26 (work in progress), October 2019.
- [I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Mattes, P., Rosen, E., Jain,
D., and S. Lin, "Advertising Segment Routing Policies in
BGP", draft-ietf-idr-segment-routing-te-policy-07 (work in
progress), July 2019.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d.,
bogdanov@google.com, b., and P. Mattes, "Segment Routing
Policy Architecture", draft-ietf-spring-segment-routing-
policy-03 (work in progress), May 2019.
- [I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca,
d., Li, C., Decraene, B., Ma, S., Yadlapalli, C.,
Henderickx, W., and S. Salsano, "Service Programming with
Segment Routing", draft-ietf-spring-sr-service-
programming-00 (work in progress), October 2019.
- [I-D.ietf0-idr-srv6-flowspec-path-redirect]
Velde, G., Patel, K., Li, Z., and H. Chen, "Flowspec
Indirection-id Redirect for SRv6", draft-ietf0-idr-srv6-
flowspec-path-redirect-02 (work in progress), July 2019.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J.,
and D. McPherson, "Dissemination of Flow Specification
Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009,
<<https://www.rfc-editor.org/info/rfc5575>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

[RFC8562] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 8562, DOI 10.17487/RFC8562, April 2019, <<https://www.rfc-editor.org/info/rfc8562>>.

Authors' Addresses

Ryo Nakamura (editor)
The University of Tokyo
Tokyo
JP

Phone: +81-3-5841-2710
Email: upa@haeena.net

Yukito Ueno
NTT Communications Corporation
Tokyo
JP

Phone: +80 90 3085 5274
Email: yukito.ueno@ntt.com

Teppei Kamata
Cisco Systems, Inc.
Tokyo
JP

Email: tkamata@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2020

D. Voyer, Ed.
Bell Canada
C. Filsfils
R. Parekh
Cisco Systems, Inc.
H. Bidgoli
Nokia
Z. Zhang
Juniper Networks
October 11, 2019

SR Replication Segment for Multi-point Service Delivery
draft-voyer-spring-sr-replication-segment-00

Abstract

This document describes the SR Replication segment for Multi-point service delivery. A SR Replication segment allows a packet to be replicated from a Replication node to downstream nodes.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 13, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--------------------------------------|---|
| 1. Introduction | 2 |
| 2. Replication segment | 2 |
| 3. IANA Considerations | 4 |
| 4. Security Considerations | 4 |
| 5. Acknowledgements | 4 |
| 6. Contributors | 4 |
| 7. Normative References | 5 |
| Authors' Addresses | 6 |

1. Introduction

We define a new type of segment for Segment Routing [RFC8402], called Replication segment, which allows a node (henceforth called as Replication node) to replicate packets to a set of other nodes (called Downstream nodes) in a Segment Routing Domain. Replication segments provide building blocks for Point-to-Multi-point Service delivery. A Replication segment at ingress node of Multi-point service replicates packets directly to each egress node of the service, without need for any state in the core of SR domain. Multiple Replication segments can be stitched together to build a tree in SR domain for Multi-point service; this is outside the scope of this document.

2. Replication segment

In a Segment Routing Domain, a Replication segment is a logical segment which connects a Replication node to a set of Downstream nodes. A Replication segment can be either provisioned locally on a node or programmed by a PCE. Replication segments apply equally to both SR-MPLS and SRv6 instantiations of Segment Routing.

A Replication segment is identified by the tuple <Replication-ID, Node-ID>, where:

- o Replication-ID: An identifier for a Replication segment that is unique in context of the Replication node. This is an unsigned 32-bit number.
- o Node-ID: The address of a node at which a Replication segment is instantiated. Replication segment is instantiated at Downstream nodes and at the Replication node.

The Replication-ID can be extended or modified as required based on specific use of a Replication segment.

A Replication segment is defined by following elements:

- o Replication SID: The Segment Identifier of a Replication Segment. This is a SR-MPLS label or a SRv6 SID [RFC8402].
- o Downstream Nodes: Set of nodes in Segment Routing domain to which a packet is replicated by the Replication segment.
- o Replication State: See below.

Replication state is a list of Replication branches to the Downstream nodes. In this document, each branch is abstracted to a <Downstream Node, Downstream Replication-SID> tuple. A Replication branch to a particular Downstream Node could be represented by the node's Node SID (i.e. it does not matter how traffic gets to the Downstream node, whether it's directly connected or not), or in case of a directly connected node it could be represented by the Adjacency SID (for the interface connecting to the directly connected Leaf Node). Alternatively, the Downstream Node could also be expanded to a SID-list that partially/fully specifies the explicit path to it. A Replication branch can also use a Segment Routing Policy [I-D.ietf-spring-segment-routing-policy], if available, from the Replication node to the Downstream node.

Replication SID identifies the Replication Segment in the forwarding plane. The Replication SID SHOULD be considered to be the equivalent of Binding SID [I-D.ietf-spring-segment-routing-policy] of a Segment Routing Policy, when Replication Segment is instantiated at Ingress node of a Multi-point service. At Downstream nodes, the Replication SID MAY be used to identify the Multi-point service.

A packet steered into a Replication Segment at a node is replicated to each Downstream node with the Downstream Replication SID that is

relevant at that node . A packet is steered into a Replication Segment in two ways:

- o Based on a local policy-based routing at Replication node.
- o When the Active Segment [RFC8402] at Replication node is the Replication SID.

3. IANA Considerations

This document makes no request of IANA.

4. Security Considerations

There are no additional security risks introduced by this design.

5. Acknowledgements

The authors would like to acknowledge Siva Sivabalan, Mike Koldychev and Vishnu Pavan Beeram for their valuable inputs.

6. Contributors

Clayton Hassen
Bell Canada
Vancouver
Canada

Email: clayton.hassen@bell.ca

Kurtis Gillis
Bell Canada
Halifax
Canada

Email: kurtis.gillis@bell.ca

Arvind Venkateswaran
Cisco Systems, Inc.
San Jose
US

Email: arvvenka@cisco.com

Zafar Ali
Cisco Systems, Inc.
US

Email: zali@cisco.com

Swadesh Agrawal
Cisco Systems, Inc.
San Jose
US

Email: swaagraw@cisco.com

Jayant Kotalwar
Nokia
Mountain View
US

Email: jayant.kotalwar@nokia.com

Tanmoy Kundu
Nokia
Mountain View
US

Email: tanmoy.kundu@nokia.com

Tarek Saad
Juniper Networks
Canada

Email: tsaad@juniper.net

7. Normative References

- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Daniel Voyer (editor)
Bell Canada
Montreal
CA

Email: daniel.voyer@bell.ca

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Rishabh Parekh
Cisco Systems, Inc.
San Jose
US

Email: riparekh@cisco.com

Hooman Bidgoli
Nokia
Ottawa
CA

Email: hooman.bidgoli@nokia.com

Zhaohui Zhang
Juniper Networks

Email: zzhang@juniper.net

Network
Internet-Draft
Intended status: Informational
Expires: May 6, 2020

C. Weiqiang
China Mobile
P. Shaofu
L. Aihua
ZTE Corporation
G. Mirsky
ZTE Corp.
W. Xiaolan
New H3C Technologies Co. Ltd
C. Wei
Centec
S. Zadok
Broadcom
November 3, 2019

Unified Identifier in IPv6 Segment Routing Networks
draft-wmsaxw-6man-usid-id-use-00

Abstract

Segment Routing architecture leverages the paradigm of source routing. It can be realized in a network data plane by prepending the packet with a list of instructions, a.k.a. segments. A segment can be encoded as a Multi-Protocol Label Switching (MPLS) label, IPv4 address, or IPv6 address. Segment Routing can be applied in the MPLS data plane by encoding segments in an MPLS label stack. It also can be applied to the IPv6 data plane by encoding a list of segment identifiers in IPv6 Segment Routing Extension Header (SRH). In this document is described the use of unified segment identifiers in use cases where interworking between SR-MPLS and SRv6 is required.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Introduction | 2 |
| 1.1. Conventions used in this document | 3 |
| 1.1.1. Terminology | 3 |
| 1.1.2. Requirements Language | 3 |
| 2. Requirements for Using SRv6 in Backhaul | 4 |
| 3. Using SRv6 U-SID in Backhaul | 4 |
| 3.1. Smoothly Upgrading to SRv6 from SR-MPLS | 4 |
| 3.2. Interworking Between SRv6 and SR-MPLS | 5 |
| 3.3. Compressing SRv6 Header Effectively | 6 |
| 3.4. Support a Super-large-scale Networking and Flexibility in Assigning Addresses | 6 |
| 4. Operations with Unified Segment Identifier | 6 |
| 5. IANA Considerations | 7 |
| 6. Security Considerations | 7 |
| 7. Acknowledgements | 7 |
| 8. Normative References | 7 |
| Authors' Addresses | 8 |

1. Introduction

Many functions related to Operation, Administration and Maintenance (OAM) require identification of the SR tunnel ingress and the path, constructed by segments, between the ingress and the egress SR nodes. Combination of IPv6 encapsulation [RFC8200] and the Source Routing Extension Header (SRH) [I-D.ietf-6man-segment-routing-header], referred to as SRv6, comply with these requirements while it is challenging when applying SR in MPLS networks [I-D.ietf-spring-segment-routing-mpls], also referred to as SR-MPLS.

On the other hand, the size of the IPv6 segment identifier (SID) presents a scaling challenge to use topological instructions that

define a strict explicitly routed path in combination with service-based instructions. At the same time, that is where the SR-MPLS approach provides better results due to smaller SID length.

SR-MPLS currently, more often than SRv6, is used in metro networks. With the gradual deployment of SRv6 in the core networks, it becomes necessary to support interworking between SR-MPLS and SRv6. Operationally it would be more efficient and straightforward if SRv6 can use the same size SIDs as in SR-MPLS. The SRH can be extended to use the same as in SR-MPLS SID length to support the unified segment identifier (U-SID) [I-D.mirsky-6man-unified-id-sr]. As a result of using this approach, U-SIDs can be used end-to-end across a tunnel that spans over SR-MPLS and SRv6 domains.

In this document is described the use of unified segment identifiers, encoded as MPLS label and/or 32 bits-long address, in use cases when interworking between SR-MPLS and SRv6 networks is required.

1.1. Conventions used in this document

1.1.1. Terminology

SR: Segment Routing

SRH: Segment Routing Extension Header

MPLS: Multiprotocol Label Switching

SR-MPLS: Segment Routing using MPLS data plane

SID: Segment Identifier

IGP: Interior Gateway Protocol

OAM: Operation, Administration and Maintenance

SRv6: Segment Routing in IPv6

U-SID: Unified Segment Identifier

1.1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Requirements for Using SRv6 in Backhaul

2G/3G/4G backhaul networks widely deploy MPLS to connect wireless services. Many operators are already deploying 5G networks. To optimize the operation of the network, many operators intent to adopt the segment routing. Currently, given maturity of SR-MPLS, it has been deployed on a large scale. Meanwhile the requirements of 5G super-large-scale number of connections accelerate the deployment of IPv6 networks. Thus, logically, operators consider SRv6 solution to fulfill the 5G backhaul requirement. But the backhaul network could not deploy SRv6 in one day, especially if it has already been using MPLS and SR-MPLS. It might be reasonable to upgrade from MPLS to SR-MPLS and then to SRv6. There are several essential operational requirements for the deployment of SRv6 in 5G backhaul network:

1. Ensure the ability to transform the existing SR-MPLS backhaul network into an SRv6 5G backhaul network incrementally.
2. Support interworking between SRv6 and SR-MPLS domains in the network.
3. Support SRv6 header compressing.
4. Support super-large-scale networking and address planning

3. Using SRv6 U-SID in Backhaul

U-SID provides a solution that complies to the 5G backhaul requirements.

3.1. Smoothly Upgrading to SRv6 from SR-MPLS

SR-MPLS uses a segment encoded as a label in an MPLS label stack to simplify the backhaul network. It leverages the advantages of both source-routing and MPLS. Existing backhaul networks that use MPLS can be first updated to use SR-MPLS. SRv6 uses the segment encoded as an identifier in IPv6 SRH. The SR-MPLS and SRv6 protocol stacks are illustrated in Figure 1.

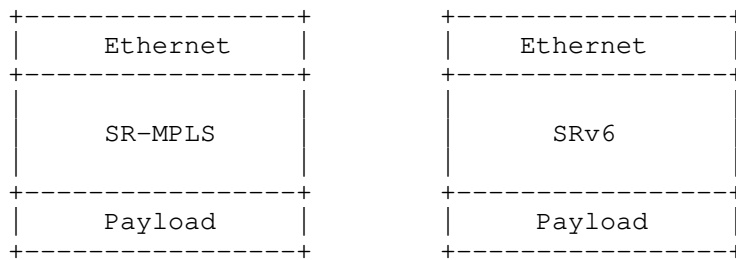


Figure 1: SR-MPLS and SRv6 Protocol Stacks

A segment identifier in SR-MPLS occupies 32 bits, and in SRv6 - 128 bits. As the backhaul infrastructure being upgraded to IPv6, operators are looking for technology that would reuse SR-MPLS by re-mapping the label table. But the namespace in SR-MPLS is limited and couldn't build the new segment identifiers to the global network. Using U-SID with SRv6 allows the reuse of the 32-bit SIDs, which are the same as in SR-MPLS. Thus, U-SID with SRv6 can be reused in backhaul to minimize the impact on existing SR-MPLS services and support smooth rollout of SRv6. The only additional task is to assign U-SIDs to the SRv6 domain. The controller could create an end-to-end SR tunnel using 32bit-long segments identifiers to stitch the SR-MPLS and SRv6 domains.

3.2. Interworking Between SRv6 and SR-MPLS

For a 5G backhaul network, the operators want to try their best to reuse the existing transport network. Consequently, they must consider the SRv6 interworking with SR-MPLS while deploying SRv6. Using U-SID offers a practical approach to native interworking between SR-MPLS and SRv6 domains because an operator in both domains can use segment identifiers of the same format, U-SID.

Using U-SID interworking between SRv6 and SR-MPLS brings some significant advantages:

1. An end-to-end LSP can be created across the access/aggregation network with SR-MPLS and core network with SRv6.
2. An end-to-end OAM and protection mechanism can be supported reusing SR-MPLS

The SR-MPLS and SRv6 interworking is illustrated in Figure 2. An end-to-end SR tunnel from A to F crosses the SR-MPLS and SRv6 domains. Using U-SID end-to-end LSP can reuse SR-MPLS forwarding, and support end-to-end OAM and protection.

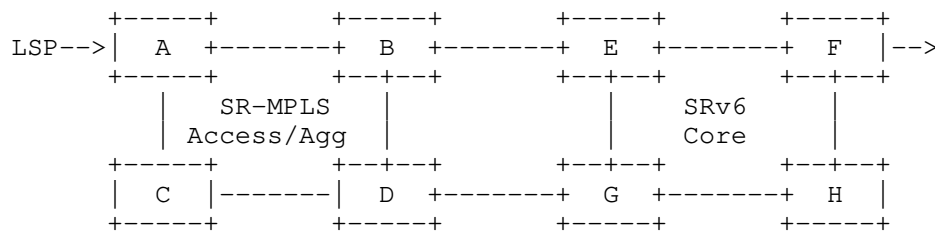


Figure 2: SR-MPLS and SRv6 Interworking

3.3. Compressing SRv6 Header Effectively

While deploying SRv6 in the backhaul network, the SRv6 header overhead must be considered. Typically there a maximum of ten hops for an end-to-end transport path. The header overhead is 1280 bits (10*128 bit SRH) using SRH with the 128-bit SID without OAM and protection. It will be reduced to 320 bits (3*128 bit SRH) using U-SID SRv6 with 32-bit SID. So the compressing rate is more than 70% (from at least 10*128 bit SRH to 3*128 bit SRH).

3.4. Support a Super-large-scale Networking and Flexibility in Assigning Addresses

The scale of the backhaul network is up to 10K nodes. A network of such size needs to support to address up to 10K nodes. U-SID SRv6 can support the 2^{20} labels as the same with MPLS, and it's enough for a super-large-scale backhaul networking. Since IPv6 solves the problem of a shortage of IPv4 addresses, it should not be using a shorter IPv6 address, i.e., a shorter prefix plus a shorter offset. That will violate the original IPv6 design. On the other hand, using SRv6 should not require the assignment of special addresses for the operator's network. U-SID can preserve the full 128-bit addresses by re-mapping the table. To use U-SID in SRv6 doesn't require the IPv6 address and SRv6 segments planning, such as the address prefix allocation. The operator would reuse the current address assignment and planning, thus minimizing the impact on the backhaul network.

4. Operations with Unified Segment Identifier

When the SRH is used to include 20-bits or 32-bits U-SIDs the ingress and transit nodes of an SR tunnel act as described in Section 5.1 and Section 5.2 of [I-D.ietf-6man-segment-routing-header] respectively.

5. IANA Considerations

This document has no requests to IANA. This section can be removed before the publication.

6. Security Considerations

This specification inherits all security considerations of [RFC8402] and [I-D.ietf-6man-segment-routing-header].

7. Acknowledgements

TBD

8. Normative References

- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.
- [I-D.mirsky-6man-unified-id-sr]
Cheng, W., Mirsky, G., Peng, S., Aihua, L., Wan, X., and C. Wei, "Unified Identifier in IPv6 Segment Routing Networks", draft-mirsky-6man-unified-id-sr-03 (work in progress), July 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Cheng Weiqiang
China Mobile
Beijing
China

Email: chengweiqiang@chinamobile.com

Peng Shaofu
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: peng.shaofu@zte.com.cn

Liu Aihua
ZTE Corporation
Zhongxing Industrial Park, Nanshan District
Shenzhen
China

Email: liu.aihua@zte.com.cn

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Wan Xiaolan
New H3C Technologies Co. Ltd
No.8, Yongjia Road, Haidian District
Beijing
China

Email: wxlan@h3c.com

Cheng Wei
Centec
Building B, No.5 Xing Han Street, Suzhou Industrial Park
Suzhou
China

Email: Chengw@centecnetworks.com

Shay
Broadcom
Israel

Email: shay.zadok@broadcom.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2020

Quan Xiong
Greg Mirsky
ZTE Corporation
Weiqiang Cheng
China Mobile
October 17, 2019

The Use of Path Segment in SR Inter-domain Scenarios
draft-xiong-spring-path-segment-sr-inter-domain-01

Abstract

This document discusses the inter-domain scenarios for SR-MPLS networks and proposes the solution with the use of path segments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 19, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Introduction | 2 |
| 2. Conventions used in this document | 3 |
| 2.1. Terminology | 3 |
| 2.2. Requirements Language | 4 |
| 3. Path Segment for SR-MPLS Inter-domain | 4 |
| 3.1. Inter-domain Path Segment | 4 |
| 3.2. End-to-end Path Segment | 4 |
| 4. SR-MPLS Inter-domain Scenarios | 5 |
| 4.1. Stitching Inter-domain with i-Path | 5 |
| 4.2. Nesting Inter-domain with e-Path | 6 |
| 5. Security Considerations | 7 |
| 6. Acknowledgements | 7 |
| 7. IANA Considerations | 8 |
| 8. Normative References | 8 |
| Authors' Addresses | 8 |

1. Introduction

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an SR Policy instantiated as an ordered list of instructions called "segments". A segment can represent any instruction, topological or service based. A segment can have a semantic local to an SR node or global within an SR domain. SR supports per-flow explicit routing while maintaining per-flow state only at the ingress nodes of the SR domain. Segment Routing can be instantiated on MPLS data plane which is referred to as SR-MPLS [I-D.ietf-spring-segment-routing-mpls]. SR-MPLS leverages the MPLS label stack to construct the SR path.

[I-D.ietf-spring-mpls-path-segment] defines a path segment identifier to support bidirectional path correlation for transport network. In the multi-domain scenarios, the SR bidirectional end-to-end tunnel MAY be established with the use of path segments. The SR-MPLS inter-domain models include the stitching and nesting inter-domain models. Path segment MAY be used to indicate the inter-domain path or the end-to-end path and correlate the inter-domain paths or end-to-end unidirectional paths to achieve the path monitoring.

As defined in [RFC8402], the headend of an SR Policy binds a Binding Segment ID (BSID) to its policy. The BSID could be bound to a SID List or selected path and used to stitch the service across multiple domains. For example, as discussed in Section 3 [I-D.ietf-spring-mpls-path-segment], the BSID can be used to identify a sub-path and stitched them to an end-to-end SR path in the nesting model. The BSID and path segment can be combined to achieve the inter-domain path monitoring. But the solution is not appropriate

for the stitching model. The policy MUST be instantiated before the end-to-end service and it can not deploy domains incrementally. Moreover, all of the BSIDs MUST be pushed onto the label stack at the headend but not all of them are popped at an edge nodes. The edge node pops one BSID and bound it to a SID List. That can not meet the independence requirement in the stitching model especially when the domains belong to different operators.

This document discusses the inter-domain scenarios for SR-MPLS networks and proposes the solution with the use of path segments for end-to-end bidirectional SR path.

2. Conventions used in this document

2.1. Terminology

ABR: Area Border Routers. Routers used to connect two IGP areas (areas in OSPF or levels in IS-IS).

A->B SID list: The SID List from SR node A to SR node B.

AS: Autonomous System. An Autonomous System is composed by one or more IGP areas.

ASBR: Autonomous System Border Router. A router used to connect together ASes of the same or different service providers via one or more inter-AS links.

BSID: Binding Segment ID.

Domains: Autonomous System (AS) or IGP Area. An Autonomous System is composed by one or more IGP areas.

e-Path: End-to-end Path Segment.

s-Path: Sub-path Path Segment.

Inter-Area: Two IGP areas interconnects with an ABR in an AS.

Inter-AS: Two ASes interconnects with an ASBR.

IGP: Interior Gateway Protocol.

i-Path/i-PSID: Inter-domain Path Segment.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Path Segment for SR-MPLS Inter-domain

3.1. Inter-domain Path Segment

In the stitching inter-domain model, the end-to-end SR path is split into multiple segments. And each segment can be identified by an inter-domain path segment (i-Path or i-PSID). The correlation of path segments can stitch the inter-domain paths and bind unidirectional paths. The i-Paths are valid in the corresponding domain and the border nodes maintain the forwarding entries of that i-Path segment, which binding with the next i-Path and SID list. At the headend node, the i-Path can correlate the inter-domain path of reverse direction and bind the two unidirectional paths. The border nodes should install the following MPLS data entries for path segments:

```
incoming label: i-Path
outgoing label: the SID list of the next domain or link + next i-Path
```

Taking Figure 1 as an example, the border node X installs the MPLS data entries:

```
incoming label: i-Path(A->X)
outgoing label: X->Y SID list + i-Path(X->Y)
```

The i-Path can be a locally unique label and assigned from the Segment Routing Local Block (SRLB). It is required that the controller (e.g., PCE) assigns the label to ensure the ingress and the egress node can recognize it and it also can be assigned from egress node of each domain. PCEP based i-Path allocation and procedure is defined in [I-D.xiong-pce-stateful-pce-sr-inter-domain].

3.2. End-to-end Path Segment

The nesting inter-domain model is described in [I-D.ietf-spring-mpls-path-segment], an end-to-end path segment, also referred to as e-Path, is used to indicate the end-to-end path, and an s-Path is used to indicate the intra-domain path. The e-Path is encapsulated at the ingress nodes and decapsulated at the egress nodes. The transit nodes, even the border nodes of domains, are not

aware of the e-Path segment. The s-Path can be used as stitching label to correlate the two domains. The use of the binding SID [RFC8402] is also recommended to reduce the size of label stack section 4.2.

The e-Path can be a globally unique or local label. If the e-Path is globally unique, it MUST be assigned from the SRGB block of each domain. If the e-Path is a local label, it is required that the controller (e.g., PCE) or a super controller (e.g., hierarchical PCE) assigns the label to ensure the ingress (A) and the egress node (Z) can recognize it and there is no SID collision in the ingress and egress domains.

4. SR-MPLS Inter-domain Scenarios

The domains of the networks may be IGP Areas or ASes and the inter-domain scenario may be inter-Area or inter-AS. The multiple SR-MPLS domains may be interconnected with a ABR within areas or inter-link between ASes. This document takes IGP Areas domains for example. SR-MPLS domains can be deployed as Figure 1 shown.

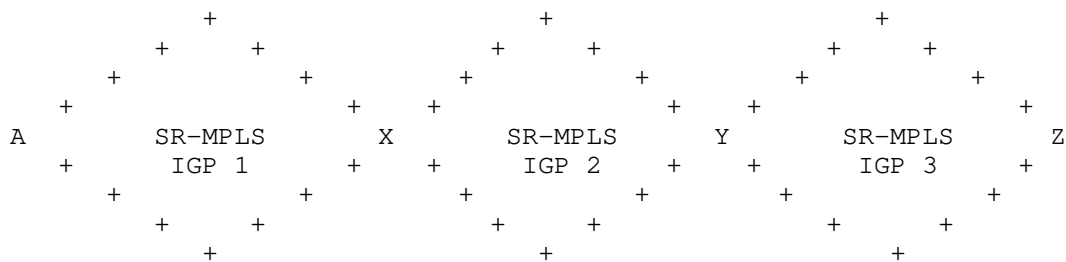


Figure 1: SR-MPLS and MPLS-TP interworking Scenario

Two SR-MPLS inter-domain models are discussed in this document including the stitching and nesting inter-domain model which are described in Section 4.1 and Section 4.2 respectively.

4.1. Stitching Inter-domain with i-Path

The Figure 1 displays the border node inter-domain scenario. SR node X and SR node Y are the border nodes of two different domains. The i-Paths from A->X, X->Y, and Y->Z are used for the inter-domain path segment. The ingress SR node A encapsulates the data packet with i-Path (A->X) and A->X SID list. The data packet is forwarded to SR node X according to the A->X SID list. Node X pushes the i-Path (X->Y) and X->Y SID list based on the above mentioned forwarding entry. The data packet is forwarded to node Y and then to the SR

node Z based on the same forwarding procedure. In node Z, the i-Path (Y->Z) can be mapped to the path from Z to Y of reverse direction and correlates the two unidirectional paths. The packet transmission of the reverse direction is the same with the forwarding direction with different i-Paths. The stitching of path segments can achieve the inter-domain stitching and path monitoring

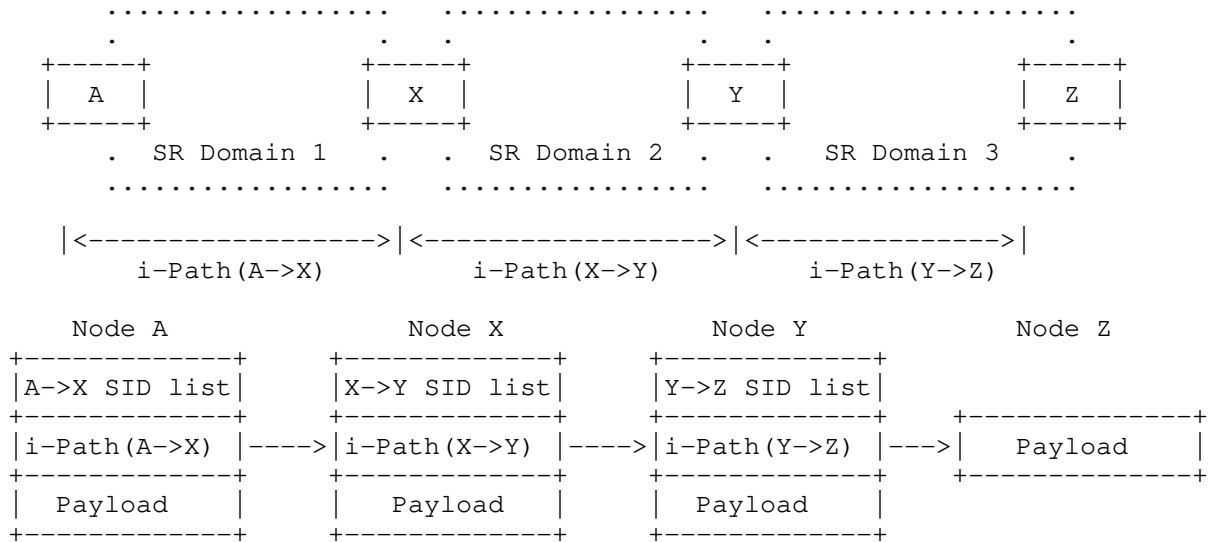


Figure 2: Stitching Border Node Inter-Domain Scenario

4.2. Nesting Inter-domain with e-Path

Figure 3 shows the SR-MPLS nesting inter-domain scenario. The e-Path(A->Z) is used to indicate the end-to-end path. The s-Path is used to identify the domain's sub-path. The e-Path, s-Path and SR list are pushed by the ingress node. The e-Path is used to correlate the two unidirectional SR paths to an SR bidirectional path. The s-Path can be used as stitching label to correlate the two inter-domain sub-paths.

The use of the binding SID [RFC8402] is also recommended to replace the SR list of each domain. As shown in Figure 3, the B-SID(X->Y) is used to replace the X->Y SID list. Ingress node A pushes e-Path(A->Z), B-SID(Y->Z), B-SID(X->Y), s-Path(A->X) and A->X SID list in turn. When the packet is received at node X, the s-Path(A->X) and X->Y SID list are popped, and the new s-Path(X->Y) is pushed. Also,

X->Y SID list replaces B-SID(X->Y) to indicate that packet to be forwarded from node X to node Y. The data packet reaches the SR node Z according to the same forwarding procedure. In SR node Z, the e-Path (A->Z) is used to correlate the two unidirectional end-to-end paths.

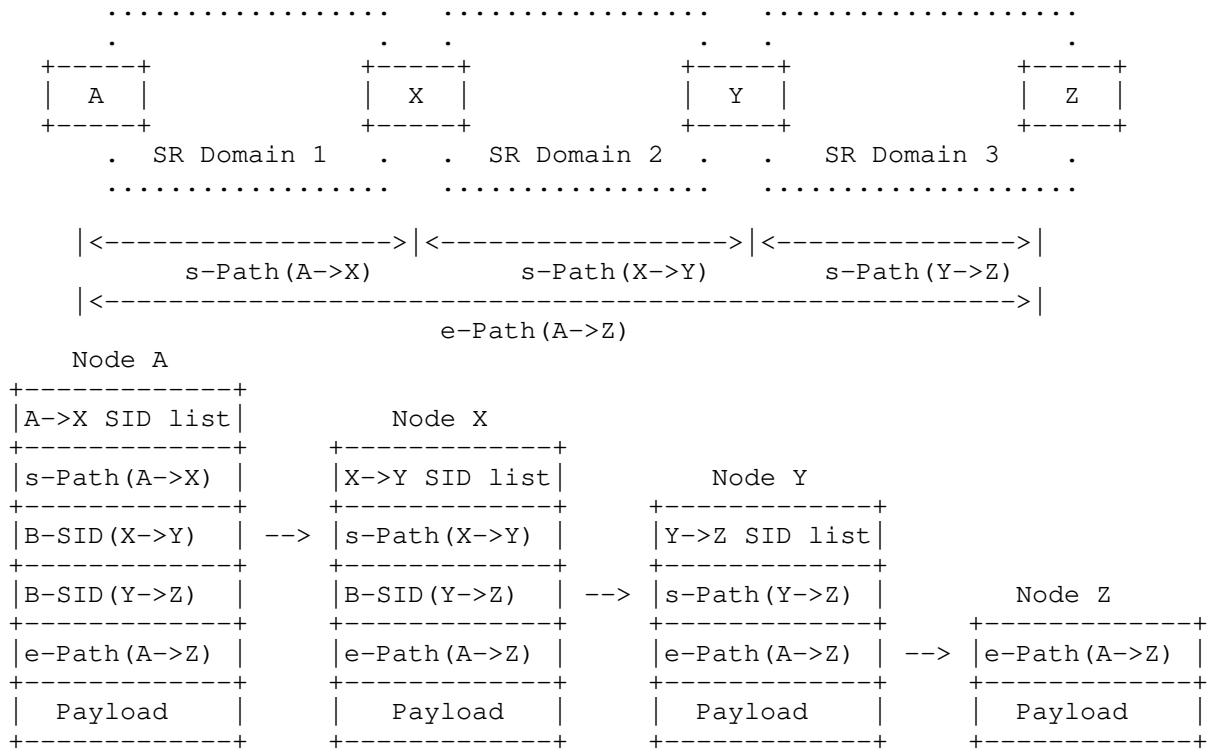


Figure 3: Nesting Inter-Domain Scenario

5. Security Considerations

TBA

6. Acknowledgements

TBA

7. IANA Considerations

TBA

8. Normative References

- [I-D.ietf-spring-mpls-path-segment]
Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler,
"Path Segment in MPLS Based Segment Routing Network",
draft-ietf-spring-mpls-path-segment-01 (work in progress),
September 2019.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-22
(work in progress), May 2019.
- [I-D.xiong-pce-stateful-pce-sr-inter-domain]
Xiong, Q., hu, f., Mirsky, G., and W. Cheng, "Stateful PCE
for SR-MPLS Inter-domain", draft-xiong-pce-stateful-pce-
sr-inter-domain-01 (work in progress), July 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Phone: +86 27 83531060
Email: xiong.quan@zte.com.cn

Greg Mirsky
ZTE Corporation
USA

Email: gregimirsky@gmail.com

Weiqiang Cheng
China Mobile
Beijing
China

Email: chengweiqiang@chinamobile.com

SPRING
Internet-Draft
Intended status: Informational
Expires: January 14, 2021

Q. Xiong
G. Mirsky
ZTE Corporation
W. Cheng
China Mobile
July 13, 2020

The Use of Path Segment in SR Inter-domain Scenarios
draft-xiong-spring-path-segment-sr-inter-domain-02

Abstract

This document illustrates the inter-domain scenarios for SR-MPLS networks to support end-to-end bidirectional tunnel across multiple domains with the use of Path Segments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Introduction | 2 |
| 2. Conventions used in this document | 3 |
| 2.1. Terminology | 3 |
| 2.2. Requirements Language | 3 |
| 3. End-to-end Path Segment | 4 |
| 3.1. S-PSID | 4 |
| 3.2. N-PSID | 4 |
| 4. SR-MPLS Inter-domain Scenarios | 4 |
| 4.1. Stitching of Path Segments | 5 |
| 4.2. Nesting of Path Segments | 6 |
| 5. Security Considerations | 7 |
| 6. Acknowledgements | 7 |
| 7. IANA Considerations | 8 |
| 8. Normative References | 8 |
| Authors' Addresses | 8 |

1. Introduction

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an SR Policy instantiated as an ordered list of instructions called "segments". A segment can represent any instruction, topological or service based. A segment can have a semantic local to an SR node or global within an SR domain. SR supports per-flow explicit routing while maintaining per-flow state only at the ingress nodes of the SR domain. Segment Routing can be instantiated on MPLS data plane which is referred to as SR-MPLS [I-D.ietf-spring-segment-routing-mpls]. SR-MPLS leverages the MPLS label stack to construct the SR path.

As defined in [RFC8402], the headend of an SR Policy binds a Binding Segment ID (B-SID) to its policy. The B-SID could be bound to a SID List or selected path and used to stitch the SR list and the SR Label Switched Paths (LSP) across multiple domains. In some scenarios, for example, a mobile backhaul transport network, it is required to provide end-to-end bidirectional path across SR networks.

[I-D.ietf-spring-mpls-path-segment] defines a path segment identifier to support bidirectional path correlation for transport network. In the multi-domain scenarios, the SR bidirectional end-to-end path MAY be established with the use of path segments. Path segment MAY be used to indicate the end-to-end bidirectional path to achieve the path monitoring including nesting of Path Segments or Path SID (N-PSID) and stitching of Path Segments or Path SID (S-PSID).

This document illustrates the inter-domain scenarios for SR-MPLS networks to support end-to-end bidirectional tunnel across multiple domains with the use of Path Segments.

2. Conventions used in this document

2.1. Terminology

ABR: Area Border Routers. Routers used to connect two IGP areas (areas in OSPF or levels in IS-IS).

A->B SID list: The SID List from SR node A to SR node B.

AS: Autonomous System. An Autonomous System is composed by one or more IGP areas.

ASBR: Autonomous System Border Router. A router used to connect together ASes of the same or different service providers via one or more inter-AS links.

B-SID: Binding Segment ID.

Domains: Autonomous System (AS) or IGP Area. An Autonomous System is composed by one or more IGP areas.

e-Path: End-to-end Path Segment.

Inter-Area: Two IGP areas interconnects with an ABR in an AS.

Inter-AS: Two ASes interconnects with an ASBR.

IGP: Interior Gateway Protocol.

N-PSID: Nesting of Path Segments.

S-PSID: Stitching of Path Segments.

SR: Segment Routing.

SR-MPLS: Segment Routing with MPLS data plane.

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. End-to-end Path Segment

3.1. S-PSID

As described in [I-D.ietf-spring-mpls-path-segment], an end-to-end Path Segment, also referred to as e-Path. In the inter-domain scenario, the end-to-end SR path is split into multiple segments. And each segment can be identified by S-PSIDs in stitching model. The correlation of path segments can stitch the inter-domain paths and bind unidirectional paths. The S-PSIDs are valid in the corresponding domain and the border nodes maintain the forwarding entries of that S-PSID. At the headend node, the S-PSID can correlate the inter-domain path of reverse direction and bind the two unidirectional paths.

The S-PSID can be a locally unique label and assigned from the Segment Routing Local Block (SRLB). It is required that the controller (e.g., PCE) assigns the label to ensure the ingress and the egress node can recognize it and it also can be assigned from egress node of each domain. PCEP based S-PSID allocation and procedure is defined in [I-D.xiong-pce-stateful-pce-sr-inter-domain].

3.2. N-PSID

As described in [I-D.ietf-spring-mpls-path-segment], an end-to-end Path Segment, also referred to as e-Path. In nesting model, the e-Path is also referred to as N-PSID which is encapsulated at the ingress nodes and decapsulated at the egress nodes. The transit nodes, even the border nodes of domains, are not aware of the N-PSID. The use of the B-SID is also recommended to reduce the size of label stack section 4.2 and stitch the SR list and the SR LSP. The N-PSID can be used to indicate the end-to-end path and achieve the bidirectional path monitoring.

The N-PSID can be a globally unique or local label. If the N-PSID is globally unique, it MUST be assigned from the SRGB block of each domain. If the N-PSID is a local label, it is required that the controller (e.g., PCE) or a super controller (e.g., hierarchical PCE) assigns the label to ensure the ingress (A) and the egress node (Z) can recognize it and there is no SID collision in the ingress and egress domains.

4. SR-MPLS Inter-domain Scenarios

The domains of the networks may be IGP Areas or ASes and the inter-domain scenario may be inter-Area or inter-AS. The multiple SR-MPLS domains may be interconnected with a ABR within areas or inter-link between ASes. This document takes IGP Areas domains for example.

The border link scenarios are in future discussion. SR-MPLS domains can be deployed as Figure 1 shown.

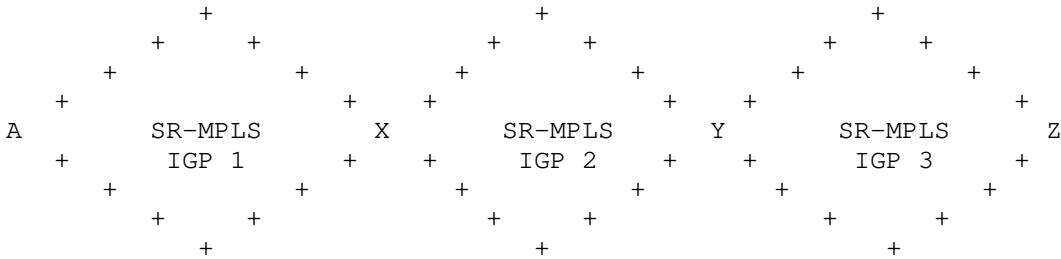


Figure 1: SR-MPLS inter-domain Scenario

Two SR-MPLS inter-domain models are discussed in this document including using the stitching and nesting of Path Segments which are described in Section 4.1 and Section 4.2 respectively.

4.1. Stitching of Path Segments

The Figure 1 displays the border node inter-domain scenario. SR node X and SR node Y are the border nodes of two different domains. The S-PSIDs from A->X, X->Y, and Y->Z are used for the inter-domain path segment. The ingress SR node A encapsulates the data packet with S-PSID (A->X), B-SID(Y->Z), B-SID(X->Y) and A->X SID list. The data packet is forwarded to SR node X according to the A->X SID list. Node X pushes the S-PSID (X->Y), B-SID(Y->Z) and X->Y SID list based on the above mentioned forwarding entry. The data packet is forwarded to node Y and then to the SR node Z based on the same forwarding procedure. In node Z, the S-PSID (Y->Z) can be mapped to the path from Z to Y of reverse direction and correlates the two unidirectional paths. The packet transmission of the reverse direction is the same with the forwarding direction with different S-PSID. The stitching of path segments can achieve the inter-domain path monitoring.

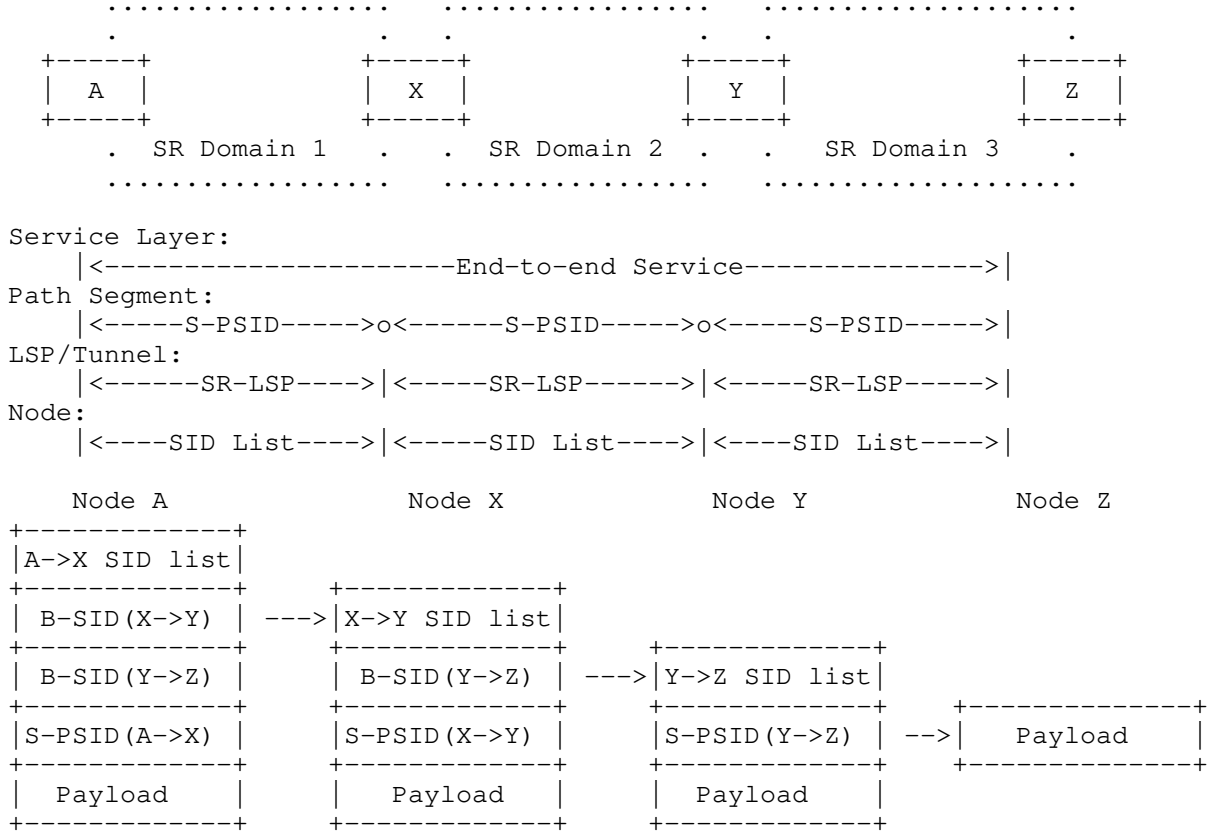


Figure 2: Stitching of Path Segments in Inter-Domain Scenario

4.2. Nesting of Path Segments

Figure 3 shows the SR-MPLS nesting inter-domain scenario. The e-Path(A->Z) is used to indicate the end-to-end path. The N-PSID, B-SID and SR list are pushed by the ingress node. The N-PSID is used to correlate the two unidirectional SR paths to an SR bidirectional path.

The use of the B-SID is also recommended to replace the SR list of each domain. As shown in Figure 3, the B-SID(X->Y) is used to replace the X->Y SID list. Ingress node A pushes N-PSID(A->Z), B-SID(Y->Z), B-SID(X->Y), and A->X SID list in turn. When the packet is received at node X, the X->Y SID list are popped. Also, X->Y SID list replaces B-SID(X->Y) to indicate that packet to be forwarded from node X to node Y. The data packet reaches the SR node Z

according to the same forwarding procedure. In SR node Z, the N-PSID (A->Z) is used to correlate the two unidirectional end-to-end paths.

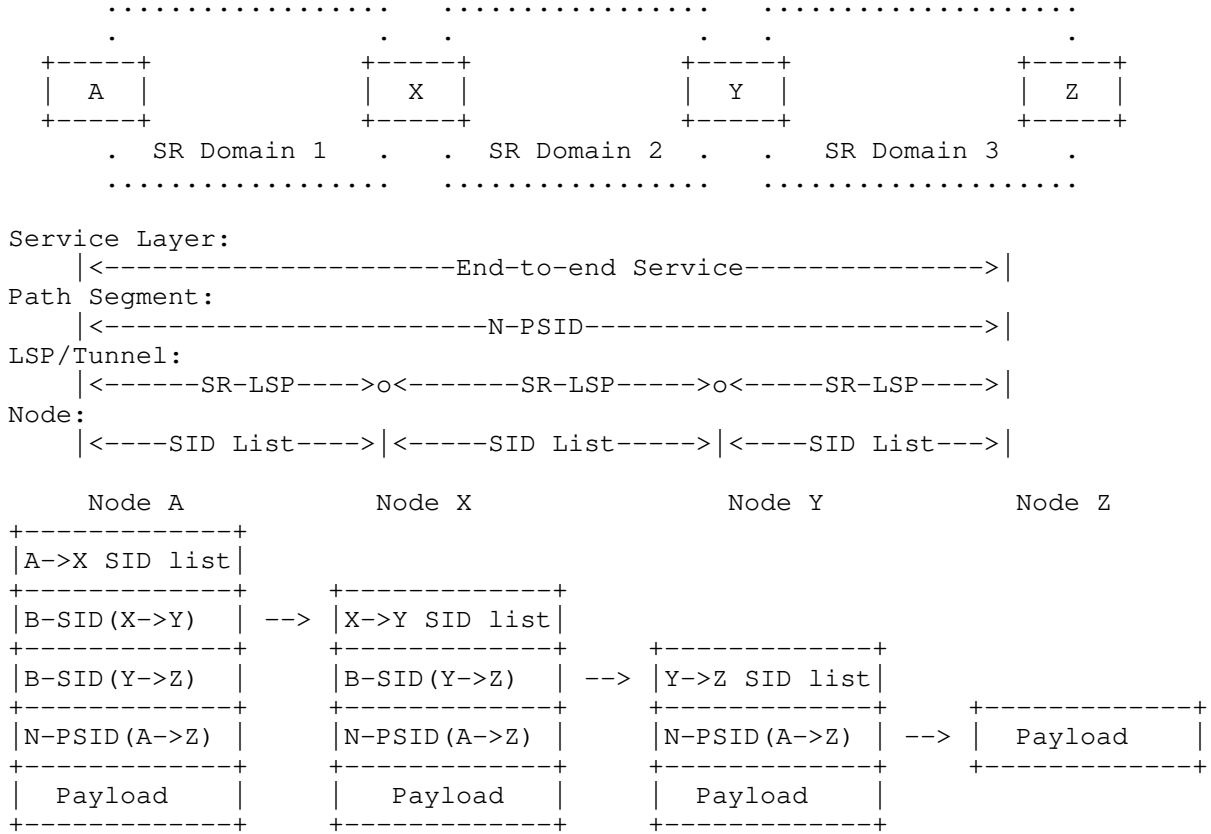


Figure 3: Nesting of Path Segments in Inter-Domain Scenario

5. Security Considerations

TBA

6. Acknowledgements

TBA

7. IANA Considerations

TBA

8. Normative References

- [I-D.ietf-spring-mpls-path-segment]
Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler,
"Path Segment in MPLS Based Segment Routing Network",
draft-ietf-spring-mpls-path-segment-02 (work in progress),
February 2020.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-22
(work in progress), May 2019.
- [I-D.xiong-pce-stateful-pce-sr-inter-domain]
Xiong, Q., Mirsky, G., hu, f., and W. Cheng, "Stateful PCE
for SR-MPLS Inter-domain", draft-xiong-pce-stateful-pce-
sr-inter-domain-02 (work in progress), October 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Phone: +86 27 83531060
Email: xiong.quan@zte.com.cn

Greg Mirsky
ZTE Corporation
USA

Email: gregimirsky@gmail.com

Weiqliang Cheng
China Mobile
Beijing
China

Email: chengweiqliang@chinamobile.com