Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2018

L. Ginsberg, Ed.
P. Psenak, Ed.
C. Filsfils
Cisco Systems
A. Bashandy
Individual
B. Decraene
Orange
Z. Hu
Huawei Technologies
June 29, 2018

IS-IS Extensions to Support Routing over IPv6 Dataplane
draft-bashandy-isis-srv6-extensions-03.txt

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths by encoding paths as sequences of topological sub-paths, called "segments". Segment routing architecture can be implemented over an MPLS data plane as well as an IPv6 data plane. This draft describes the IS-IS extensions required to support Segment Routing over an IPv6 data plane.

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 December 31, 2018.

Table of Contents

1. Introduction

With Segment Routing (SR) [I-D.ietf-spring-segment-routing], a node steers a packet through an ordered list of instructions, called segments.

Segments are identified through Segment Identifiers (SIDs).

Segment Routing can be directly instantiated on the IPv6 data plane through the use of the Segment Routing Header defined in [I-D.ietf-6man-segment-routing-header]. SRv6 refers to this SR instantiation on the IPv6 dataplane.

The network programming paradigm [I-D.filsfils-spring-srv6-network-programming] is central to SRv6. It describes how any function can be bound to a SID and how any network program can be expressed as a combination of SID’s.

This document specifies IS-IS extensions that allow the IS-IS protocol to encode some of these functions.

Familiarity with the network programming paradigm [I-D.filsfils-spring-srv6-network-programming] is necessary to understand the extensions specified in this document.

This document defines one new top level IS-IS TLV and several new IS-IS sub-TLVs.

The SRv6 Capabilities sub-TLV announces the ability to support SRv6 and some Endpoint functions listed in Section 7 as well as advertising limitations when applying such Endpoint functions.

The SRv6 Locator top level TLV announces SRv6 locators - a form of summary address for the set of topology/algorithm specific SIDs associated with a node.

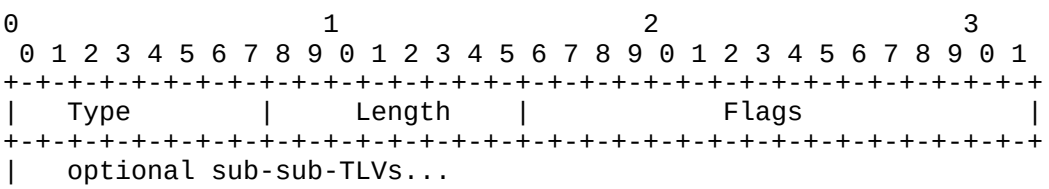
The SRv6 End SID sub-TLV, the SRv6 End.X SID sub-TLV, and the SRv6 LAN End.X SID sub-TLV are used to advertise which SIDs are instantiated at a node and what Endpoint function is bound to each instantiated SID.

2. SRv6 Capabilities sub-TLV

A node indicates that it has support for SRv6 by advertising a new SRv6- capabilities sub-TLV of the router capabilities TLV [RFC7981].

The SRv6 Capabilities sub-TLV may contain optional sub-sub-TLVs. No sub-sub-TLVs are currently defined.

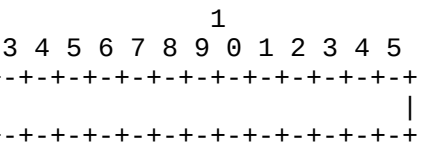
The SRv6 Capabilities sub-TLV has the following format:



Type: Suggested value 25, to be assigned by IANA

Length: 2 + length of sub-sub-TLVs

Flags: 2 octets The following flags are defined:



where:

O-flag: If set, the router supports use of the O-bit in the Segment Routing Header(SRH) as defined in [I-D.ali-spring-srv6-oam].

3. Advertising Maximum SRv6 SID Depths

[I-D.ietf-isis-segment-routing-msd] defines the means to advertise node/link specific values for Maximum SID Depths (MSD) of various types. Node MSDs are advertised in a sub-TLV of the Router Capabilities TLV [RFC7981]. Link MSDs are advertised in a sub-TLV of TLVs 22, 23, 141, 222, and 223.

This document defines the relevant SRv6 MSDs and requests MSD type assignments in the MSD Types registry created by [I-D.ietf-isis-segment-routing-msd].

3.1. Maximum Segments Left MSD Type

The Maximum Segments Left MSD Type specifies the maximum value of the "SL" field [I-D.ietf-6man-segment-routing-header] in the SRH of a received packet before applying the Endpoint function associated with a SID.

SRH Max SL Type: 41 (Suggested value - to be assigned by IANA)

If no value is advertised the supported value is assumed to be 0.

3.2. Maximum End Pop MSD Type

The Maximum End Pop MSD Type specifies the maximum number of SIDs in the top SRH in an SRH stack to which the router can apply "PSP" or USP" as defined in [I-D.filsfils-spring-srv6-network-programming] flavors.

SRH Max End Pop Type: 42 (Suggested value - to be assigned by IANA)

If the advertised value is zero or no value is advertised then it is assumed that the router cannot apply PSP or USP flavors.

3.3. Maximum T.Insert MSD Type

The Maximum T.Insert MSD Type specifies the maximum number of SIDs that can be inserted as part of the "T.insert" behavior as defined in [I-D.filsfils-spring-srv6-network-programming].

SRH Max T.insert Type: 43 (Suggested value - to be assigned by IANA)

If the advertised value is zero or no value is advertised then the router is assumed not to support any variation of the "T.insert" behavior.

3.4. Maximum T.Encaps MSD Type

The Maximum T.Encaps MSD Type specifies the maximum number of SIDs that can be included as part of the "T.Encaps" behavior as defined in [I-D.filsfils-spring-srv6-network-programming] .

SRH Max T.encaps Type: 44 (Suggested value - to be assigned by IANA)

If the advertised value is zero then the router can apply T.Encaps only by encapsulating the incoming packet in another IPv6 header without SRH the same way IPinIP encapsulation is performed.

If the advertised value is non-zero then the router supports both IPinIP and SRH encapsulation subject to the SID limitation specified by the advertised value.

3.5. Maximum End D MSD Type

The Maximum End D MSD Type specifies the maximum number of SIDs in an SRH when performing decapsulation associated with "End.Dx" functions (e.g., "End.DX6" and "End.DT6") as defined in [I-D.filsfils-spring-srv6-network-programming].

SRH Max End D Type: 45 (Suggested value - to be assigned by IANA)

If the advertised value is zero or no value is advertised then it is assumed that the router cannot apply "End.DX6" or "End.DT6" functions if the extension header right underneath the outer IPv6 header is an SRH.

4. SRv6 SIDs and Reachability

As discussed in [I-D.filsfils-spring-srv6-network-programming], an SRv6 Segment Identifier (SID) is 128 bits and represented as

LOC:FUNCT

where LOC (the locator portion) 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 operator is free to use the locator length it chooses.

A node is provisioned with topology/algorithm specific locators for each of the topology/algorithm pairs supported by that node. Each locator is a covering prefix for all SIDs provisioned on that node which have the matching topology/algorithm.

Locators MUST be advertised in the SRv6 Locator TLV (see Section 6.1). Forwarding entries for the locators advertised in the SRv6 Locator TLV MUST be installed in the forwarding plane of receiving SRv6 capable routers when the associated topology/algorithm is supported by the receiving node.

Locators are routable and MAY also be advertised in Prefix Reachability TLVs (236 or 237).

Locators associated with algorithm 0 (for all supported topologies) SHOULD be advertised in a Prefix Reachability TLV (236 or 237) so that legacy routers (i.e., routers which do NOT support SRv6) will install a forwarding entry for algorithm 0 SRv6 traffic.

In cases where a locator advertisement is received in both in a Prefix Reachability TLV and an SRv6 Locator TLV, the Prefix Reachability advertisement MUST be preferred when installing entries

in the forwarding plane. This is to prevent inconsistent forwarding entries on SRv6 capable/SRv6 incapable routers.

SRv6 SIDs are advertised as sub-TLVs in the SRv6 Locator TLV except for SRv6 End.X SIDs/LAN End.X SIDs which are associated with a specific Neighbor/Link and are therefore advertised as sub-TLVs in TLVs 22, 23, 222, 223, and 141.

SRv6 SIDs are not directly routable and MUST NOT be installed in the forwarding plane. Reachability to SRv6 SIDs depends upon the existence of a covering locator.

Adherence to the rules defined in this section will assure that SRv6 SIDs associated with a supported topology/algorithm pair will be forwarded correctly, while SRv6 SIDs associated with an unsupported topology/algorithm pair will be dropped. NOTE: The drop behavior depends on the absence of a default/summary route covering a given locator.

In order for forwarding to work correctly, the locator associated with SRv6 SID advertisements MUST be the longest match prefix installed in the forwarding plane for those SIDs. There are a number of ways in which this requirement could be compromised

- Another locator associated with a different topology/algorithm is the longest match
- A prefix advertisement (i.e., from TLV 236 or 237) is the longest match

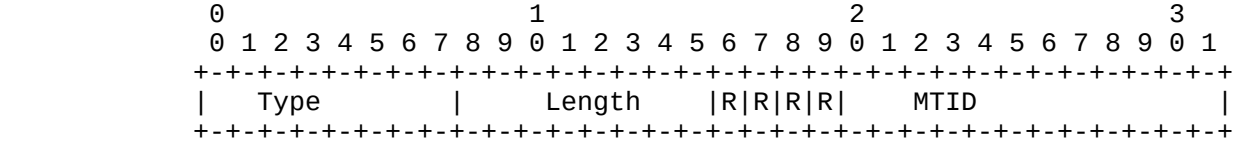
5. Advertising Locators and End SIDs

The SRv6 Locator TLV is introduced to advertise SRv6 Locators and End SIDs associated with each locator.

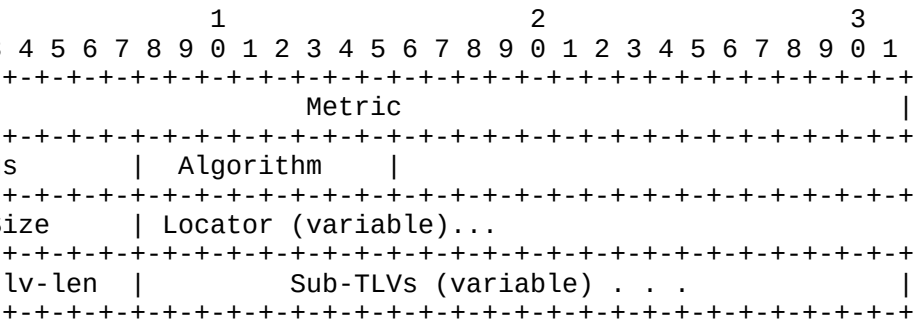
This new TLV shares the sub-TLV space defined for TLVs 135, 235, 236 and 237.

5.1. SRv6 Locator TLV Format

The SRv6 Locator TLV has the following format:



Followed by one or more locator entries of the form:



Type: 27 (Suggested value to be assigned by IANA)

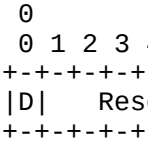
Length: variable.

MTID: Multitopology Identifier as defined in [RFC5120].
Note that the value 0 is legal.

Locator entry:

Metric: 4 octets. As described in [RFC5305].

Flags: 1 octet. The following flags are defined



where:
D bit: When the Locator is leaked from level-2 to level-1, the D bit MUST be set. Otherwise, this bit MUST be clear. Locators with the D bit set MUST NOT be leaked from level-1 to level-2. This is to prevent looping.

The remaining bits are reserved for future use. They SHOULD be set to zero on transmission and MUST be ignored on receipt.

Algorithm: 1 octet. Associated algorithm. Algorithm values are defined in the IGP Algorithm Type registry.

Loc-Size: 1 octet. Number of bits in the Locator field.
(1 - 128)

Locator: 1-16 octets. This field encodes the advertised SRv6 Locator. The Locator is encoded in the minimal number of octets for the given number of bits.

Sub-TLV-length: 1 octet. Number of octets used by sub-TLVs

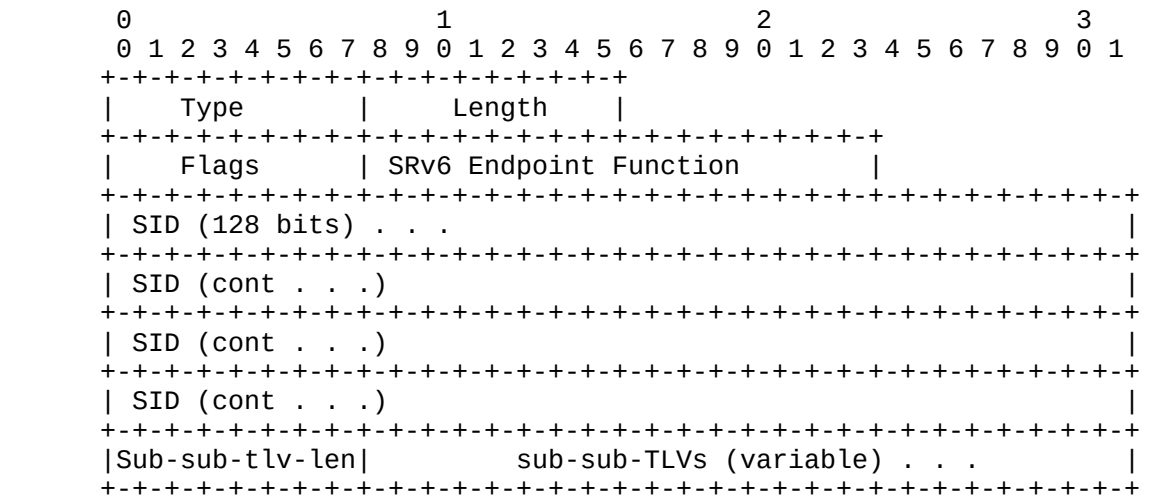
Optional sub-TLVs.

5.2. SRv6 End SID sub-TLV

The SRv6 End SID sub-TLV is introduced to advertise SRv6 Segment Identifiers (SID) with Endpoint functions which do not require a particular neighbor in order to be correctly applied [I-D.filsfils-spring-srv6-network-programming]. SRv6 SIDs associated with a neighbor are advertised using the sub-TLVs defined in Section 6.

This new sub-TLV is advertised in the SRv6 Locator TLV defined in the previous section. SRv6 End SIDs inherit the topology/algorithm from the parent locator.

The SRv6 End SID sub-TLV has the following format:



Type: 5 (Suggested value to be assigned by IANA)

Length: variable.

Flags: 1 octet. No flags are currently defined.

SRv6 Endpoint Function: 2 octets. As defined in
[I-D.filsfils-spring-srv6-network-programming]
Legal function values for this sub-TLV are defined in Section 7.

SID: 16 octets. This field encodes the advertised SRv6 SID.

Sub-sub-TLV-length: 1 octet. Number of octets used by sub-sub-TLVs

Optional sub-sub-TLVs

The SRv6 End SID MUST be a subnet of the associated Locator. SRv6 End SIDs which are NOT a subnet of the associated locator MUST be ignored.

Multiple SRv6 End SIDs MAY be associated with the same locator. In cases where the number of SRv6 End SID sub-TLVs exceeds the capacity of a single TLV, multiple Locator TLVs for the same locator MAY be advertised. For a given MTID/Locator the algorithm MUST be the same in all TLVs. If this restriction is not met all TLVs for that MTID/ Locator MUST be ignored.

6. Advertising SRv6 End.X SIDs

Certain SRv6 Endpoint functions [I-D.filsfils-spring-srv6-network-programming] must be associated with a particular neighbor, and in case of multiple layer 3 links to the same neighbor, with a particular link in order to be correctly applied.

This document defines two new sub-TLVs of TLV 22, 23, 222, 223, and 141 - namely "SRv6 End.X SID" and "SRv6 LAN End.X SID".

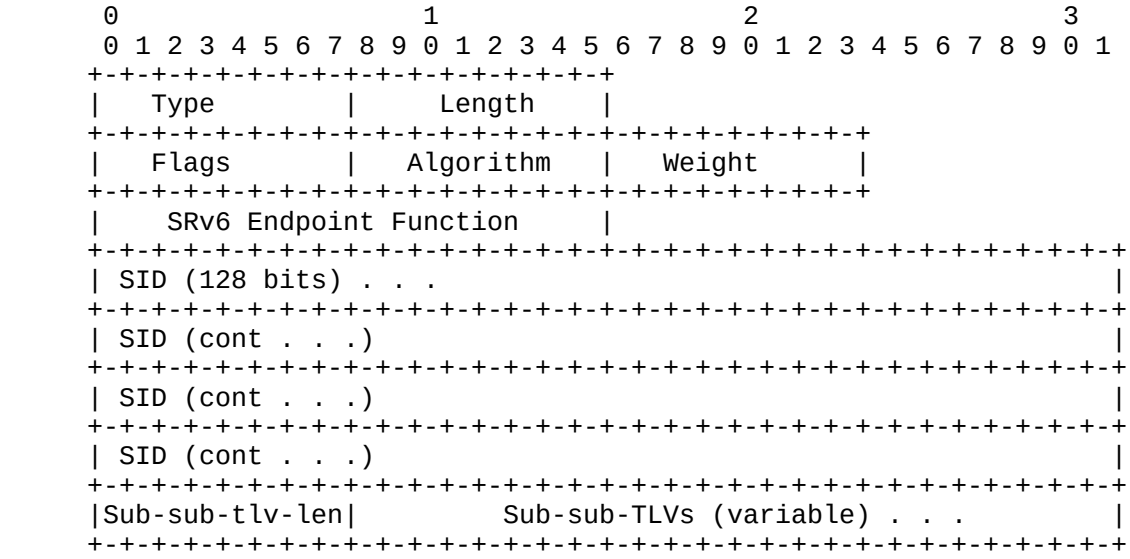
IS-IS Neighbor advertisements are topology specific - but not algorithm specific. End.X SIDs therefore inherit the topology from the associated neighbor advertisement, but the algorithm is specified in the individual SID.

All End.X SIDs MUST be a subnet of a Locator with matching topology and algorithm which is advertised by the same node in an SRv6 Locator TLV. End.X SIDs which do not meet this requirement MUST be ignored.

6.1. SRv6 End.X SID sub-TLV

This sub-TLV is used to advertise an SRv6 SID associated with a point to point adjacency. Multiple SRv6 End.X SID sub-TLVs MAY be associated with the same adjacency.

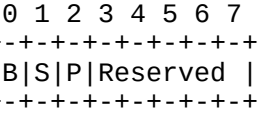
The SRv6 End.X SID sub-TLV has the following format:



Type: 43 (Suggested value to be assigned by IANA)

Length: variable.

Flags: 1 octet.



where:

B-Flag: Backup flag. If set, the End.X SID is eligible for protection (e.g., using IPFRR) as described in [RFC8355].

S-Flag. Set flag. When set, the S-Flag indicates that the End.X SID refers to a set of adjacencies (and therefore MAY be assigned to other adjacencies as well).

P-Flag. Persistent flag. When set, the P-Flag indicates that the End.X SID is persistently allocated, i.e., the End.X SID value remains consistent across router restart and/or interface flap.

Other bits: MUST be zero when originated and ignored when received.

Algorithm: 1 octet. Associated algorithm. Algorithm values are defined in the IGP Algorithm Type registry.

Weight: 1 octet. The value represents the weight of the End.X SID for the purpose of load balancing. The use of the weight is defined in [I-D.ietf-spring-segment-routing].

SRv6 Endpoint Function: 2 octets. As defined in [I-D.filsfils-spring-srv6-network-programming]
Legal function values for this sub-TLV are defined in Section 7.

SID: 16 octets. This field encodes the advertised SRv6 SID.

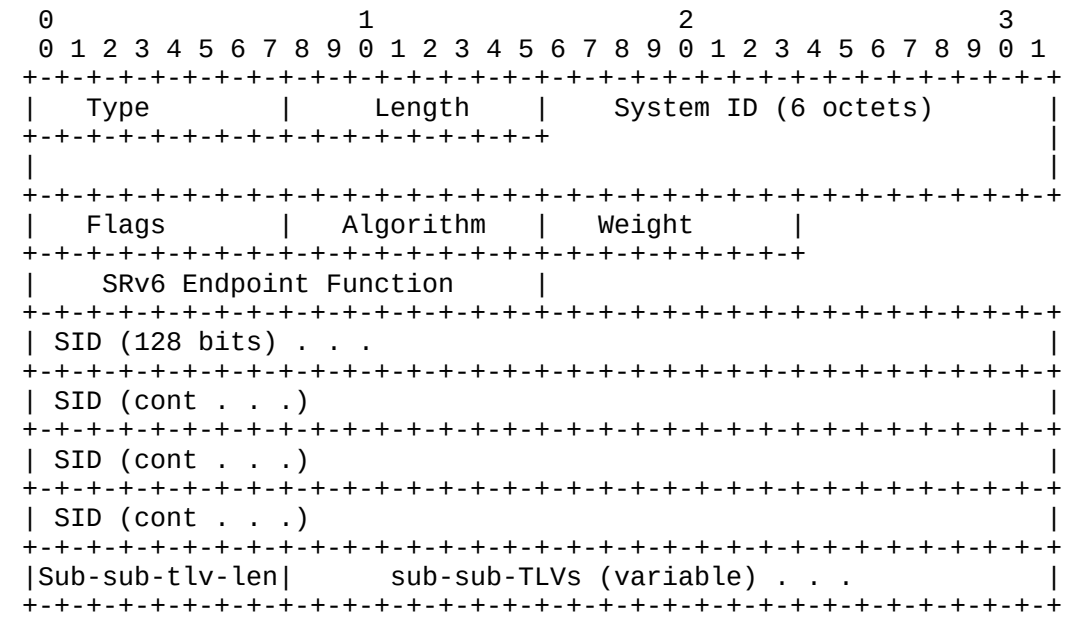
Sub-sub-TLV-length: 1 octet. Number of octets used by sub-sub-TLVs

Note that multiple TLVs for the same neighbor may be required in order to advertise all of the SRv6 End.X SIDs associated with that neighbor.

6.2. SRv6 LAN End.X SID sub-TLV

This sub-TLV is used to advertise an SRv6 SID associated with a LAN adjacency. Since the parent TLV is advertising an adjacency to the Designated Intermediate System(DIS) for the LAN, it is necessary to include the System ID of the physical neighbor on the LAN with which the SRv6 SID is associated. Given that a large number of neighbors may exist on a given LAN a large number of SRv6 LAN END.X SID sub-TLVs may be associated with the same LAN. Note that multiple TLVs for the same DIS neighbor may be required in order to advertise all of the SRv6 End.X SIDs associated with that neighbor.

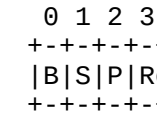
The SRv6 LAN End.X SID sub-TLV has the following format:



Type: 44 (Suggested value to be assigned by IANA)
Length: variable.

System-ID: 6 octets of IS-IS System-ID of length "ID Length" as defined in [ISO10589].

Flags: 1 octet.



where B,S, and P flags are as described in Section 6.1.
Other bits: MUST be zero when originated and ignored when received.

Algorithm: 1 octet. Associated algorithm. Algorithm values are defined in the IGP Algorithm Type registry.

Weight: 1 octet. The value represents the weight of the End.X SID for the purpose of load balancing. The use of the weight is defined in [I-D.ietf-spring-segment-routing].

SRv6 Endpoint Function: 2 octets. As defined in [I-D.filsfils-spring-srv6-network-programming]
Legal function values for this sub-TLV are defined in Section 7.

SID: 16 octets. This field encodes the advertised SRv6 SID.

Sub-sub-TLV-length: 1 octet. Number of octets used by sub-sub-TLVs.

7. Advertising Endpoint Function Types

Endpoint function types are defined in [I-D.filsfils-spring-srv6-network-programming]. The numerical values are defined in the "SRv6 Endpoint Types" registry defined in [I-D.filsfils-spring-srv6-network-programming]. This section lists the Endpoint function types which MAY be advertised by IS-IS and the SID sub-TLVs in which each type MAY appear.

Endpoint Function Type	End SID	End.X SID	Lan End.X SID
End(no PSP, no USP)	Y	N	N
End(with PSP)	Y	N	N
End(with USP)	Y	N	N
End(with PSP & USP)	Y	N	N
End.T(no PSP, no USP)	Y	N	N
End.T(with PSP)	Y	N	N
End.T(with USP)	Y	N	N
End.T(with PSP & USP)	Y	N	N
End.X(no PSP, no USP)	N	Y	Y
End.X(with PSP)	N	Y	Y
End.X(with USP)	N	Y	Y
End.X(with PSP & USP)	N	Y	Y
End.DX6	N	Y	Y
End.DT6	Y	N	N
End.OTP	Y	N	N

8. IANA Considerations

This document requests allocation for the following TLVs, sub- TLVs, and sub-sub-TLVs as well updating the ISIS TLV registry and defining a new registry.

8.1. SRv6 Locator TLV

This document adds one new TLV to the IS-IS TLV Codepoints registry.

Value: 27 (suggested - to be assigned by IANA)

Name: SRv6 Locator

This TLV shares sub-TLV space with existing "Sub-TLVs for TLVs 135, 235, 236 and 237 registry". The name of this registry needs to be changed to "Sub-TLVs for TLVs 27, 135, 235, 236 and 237 registry".

8.1.1. SRv6 End SID sub-TLV

This document adds the following new sub-TLV to the (renamed) "Sub-TLVs for TLVs 27, 135, 235, 236 and 237 registry".

Value: 5 (suggested - to be assigned by IANA)

Name: SRv6 End SID

This document requests the creation of a new IANA managed registry for sub-sub-TLVs of the SRv6 End SID sub-TLV. The registration procedure is "Expert Review" as defined in [RFC7370]. Suggested registry name is "sub-sub-TLVs for SRv6 End SID sub-TLV". No sub-sub-TLVs are defined by this document except for the reserved value.

0: Reserved

1-255: Unassigned

8.1.2. Revised sub-TLV table

The revised table of sub-TLVs for the (renamed) "Sub-TLVs for TLVs 27, 135, 235, 236 and 237 registry" is shown below:

Type	27	135	235	236	237
1	n	y	y	y	y
2	n	y	y	y	y
3	n	y	y	y	y
4	y	y	y	y	y
5	y	n	n	n	n
11	y	y	y	y	y
12	y	y	y	y	y

8.2. SRv6 Capabilities sub-TLV

This document adds the definition of a new sub-TLV in the "Sub- TLVs for TLV 242 registry".

Type: 25 (Suggested - to be assigned by IANA)

Description: SRv6 Capabilities

This document requests the creation of a new IANA managed registry for sub-sub-TLVs of the SRv6 Capability sub-TLV. The registration procedure is "Expert Review" as defined in [RFC7370]. Suggested registry name is "sub-sub-TLVs for SRv6 Capability sub-TLV". No sub-sub-TLVs are defined by this document except for the reserved value.

0: Reserved

1-255: Unassigned

8.3. SRv6 End.X SID and SRv6 LAN End.X SID sub-TLVs

This document adds the definition of two new sub-TLVs in the "sub-TLVs for TLV 22, 23, 25, 141, 222 and 223 registry".

Type: 43 (suggested - to be assigned by IANA)

Description: SRv6 End.X SID

Type: 44 (suggested - to be assigned by IANA)

Description: SRv6 LAN End.X SID

Type	22	23	25	141	222	223
43	y	y	y	y	y	y
44	y	y	y	y	y	y

8.4. MSD Types

This document defines the following new MSD types. These types are to be defined in the IGP MSD Types registry defined in [I-D.ietf-isis-segment-routing-msd] .

All values are suggested values to be assigned by IANA.

Type	Description
41	SRH Max SL
42	SRH Max End Pop
43	SRH Max T.insert
44	SRH Max T.encaps
45	SRH Max End D

9. Security Considerations

Security concerns for IS-IS are addressed in [ISO10589], [RFC5304], and [RFC5310].

10. Contributors

The following people gave a substantial contribution to the content of this document and should be considered as co-authors:

Stefano Previdi
Huawei Technologies
Email: stefano@previdi.net

Paul Wells
Cisco Systems
Saint Paul,
Minnesota
United States
Email: pauwells@cisco.com

Daniel Voyer
Email: daniel.voyer@bell.ca

Satoru Matsushima
Email: satoru.matsushima@g.softbank.co.jp

Bart Peirens
Email: bart.peirens@proximus.com

Hani Elmalky
Email: hani.elmalky@ericsson.com

Prem Jonnalagadda
Email: prem@barefootnetworks.com

Milad Sharif
Email: msharif@barefootnetworks.com>

Robert Hanzl
Cisco Systems
Millenium Plaza Building, V Celnici 10, Prague 1,
Prague, Czech Republic
Email rhanzl@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.
Email: ketant@cisco.com

11. References

11.1. Normative References

[I-D.ali-spring-srv6-oam]
Ali, Z., Filsfils, C., Kumar, N., Pignataro, C., faiqbal@cisco.com, f., Gandhi, R., Leddy, J., Matsushima, S., Raszuk, R., Peirens, B., and G. Naik, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ali-spring-srv6-oam-00 (work in progress), February 2018.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Li, Z., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., and M. Sharif, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-04 (work in progress), March 2018.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-12 (work in progress), May 2018.

[ISO10589]
"Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473), ISO/IEC 10589:2002, Second Edition.", Nov 2002.

[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>.

[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>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[RFC7370] Ginsberg, L., "Updates to the IS-IS TLV Codepoints Registry", RFC 7370, DOI 10.17487/RFC7370, September 2014, <https://www.rfc-editor.org/info/rfc7370>.

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <https://www.rfc-editor.org/info/rfc7981>.

[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-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.

[RFC8355] Filsfils, C., Ed., Previdi, S., Ed., Decraene, B., and R. Shakir, "Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks", RFC 8355, DOI 10.17487/RFC8355, March 2018, <https://www.rfc-editor.org/info/rfc8355>.

Authors' Addresses

Les Ginsberg (editor)
Cisco Systems
821 Alder Drive
Milpitas, CA 95035
USA

Email: ginsberg@cisco.com

Peter Psenak (editor)
Cisco Systems
Pribinova Street 10
Bratislava 81109
Slovakia

Email: ppsenak@cisco.com

Clarence Filsfils
Cisco Systems
Brussels
Belgium

Email: cfilsfil@cisco.com

Ahmed Bashandy
Individual

Email: abashandy.ietf@gmail.com

Bruno Decraene
Orange
Issy-les-Moulineaux
France

Email: bruno.decraene@orange.com

Zhibo Hu
Huawei Technologies

Email: huzhibo@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 24, 2019

H. Chen
D. Cheng
Huawei Technologies
M. Toy
Verizon
Y. Yang
IBM

September 20, 2018

LS Flooding Reduction
draft-cc-ospf-flooding-reduction-04

Abstract

This document proposes an approach to flood link states on a topology that is a subgraph of the complete topology per underline physical network, so that the amount of flooding traffic in the network is greatly reduced, and it would reduce convergence time with a more stable and optimized routing environment. The approach can be applied to any network topology in a single area.

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 March 24, 2019.

Table of Contents

1. Introduction

For some networks such as dense Data Center (DC) networks, the existing Link State (LS) flooding mechanism is not efficient and may have some issues. The extra LS flooding consumes network bandwidth. Processing the extra LS flooding, including receiving, buffering and decoding the extra LSs, wastes memory space and processor time. This may cause scalability issues and affect the network convergence negatively.

This document proposes an approach to minimize the amount of flooding traffic in the network. Thus the workload for processing the extra LS flooding is decreased significantly. This would improve the scalability, speed up the network convergence, stable and optimize the routing environment.

This approach is also flexible. It has multiple modes for computation of flooding topology. Users can select a mode they prefer, and smoothly switch from one mode to another. The approach is applicable to any network topology in a single area. It is backward compatible.

2. Terminology

Flooding Topology:
: A sub-graph or sub-network of a given (physical) network topology that has the same reachability to every node as the given network topology, through which link states are flooded.

critical link or interface on a flooding topology:
: A only link or interface among some nodes on the flooding topology. When this link or interface goes down, the flooding topology will be split.

critical node on a flooding topology:

A only node connecting some nodes on the flooding topology. When this node goes down, the flooding topology will be split.

backup path:
A path or a sequence of links, when a critical link or node goes down, providing a connection to connect two parts of a split flooding topology. When a critical node goes down, the flooding topology may be split into more than two parts. In this case, two or more backup paths are needed to connect all the split parts into one.

Remaining Flooding Topology:
A topology from a flooding topology by removing the failed links and nodes from the flooding topology.

LSA:
A Link State Advertisement in OSPF.

LSP:
A Link State Protocol Data Unit (PDU) in IS-IS.

LS:
A Link Sate, which is an LSA or LSP.

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. Problem Statement

OSPF and IS-IS deploy a so-called reliable flooding mechanism, where a node must transmit a received or self-originated LS to all its interfaces (except the interface where an LS is received). While this mechanism assures each LS being distributed to every node in an area or domain, the side-effect is that the mechanism often causes redundant LS, which in turn forces nodes to process identical LS more than once. This results in the waste of link bandwidth and nodes' computing resources, and the delay of topology convergence.

This becomes more serious in networks with large number of nodes and links, and in particular, higher degree of interconnection (e.g., meshed topology, spine-leaf topology, etc.). In some environments such as in data centers, the drawback of the existing flooding mechanism has already caused operational issues, including repeated and waves of flooding storms, chock of computing resources, slow

convergence, oscillating topology changes, instability of routing environment.

One example is as shown in Figure 1, where Node 1, Node 2 and Node 3 are interconnected in a mesh. When Node 1 receives a new or updated LS on its interface I11, it by default would forward the LS to its interface Il2 and I13 towards Node 2 and Node 3, respectively, after processing. Node 2 and Node 3 upon reception of the LS and after processing, would potentially flood the same LS over their respective interface I23 and I32 toward each other, which is obviously not necessary and at the cost of link bandwidth as well as both nodes' computing resource.

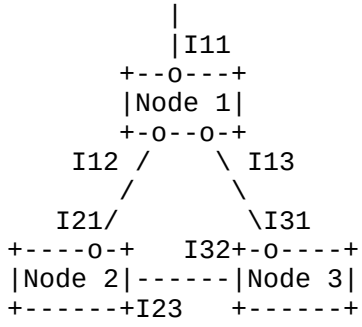


Figure 1

5. Flooding Topology

For a given network topology, a flooding topology is a sub-graph or sub-network of the given network topology that has the same reachability to every node as the given network topology. Thus all the nodes in the given network topology MUST be in the flooding topology. All the nodes MUST be inter-connected directly or indirectly. As a result, LS flooding will in most cases occur only on the flooding topology, that includes all nodes but a subset of links. Note even though the flooding topology is a sub-graph of the original topology, any single LS MUST still be disseminated in the entire network.

5.1. Construct Flooding Topology

Many different flooding topologies can be constructed for a given network topology. A chain connecting all the nodes in the given network topology is a flooding topology. A circle connecting all the nodes is another flooding topology. A tree connecting all the nodes is a flooding topology. In addition, the tree plus the connections

between some leaves of the tree and branch nodes of the tree is a flooding topology.

The following parameters need to be considered for constructing a flooding topology:

- Number of links: The number of links on the flooding topology is a key factor for reducing the amount of LS flooding. In general, the smaller the number of links, the less the amount of LS flooding.

- Diameter: The shortest distance between the two most distant nodes on the flooding topology is a key factor for reducing the network convergence time. The smaller the diameter, the less the convergence time.

- Redundancy: The redundancy of the flooding topology means a tolerance to the failures of some links and nodes on the flooding topology. If the flooding topology is split by some failures, it is not tolerant to these failures. In general, the larger the number of links on the flooding topology is, the more tolerant the flooding topology to failures.

There are many different ways to construct a flooding topology for a given network topology. A few of them are listed below:

- Central Mode: One node in the network builds a flooding topology and floods the flooding topology to all the other nodes in the network (This seems not good. Flooding the flooding topology may increase the flooding. The amount of traffic for flooding the flooding topology should be minimized.);

- Distributed Mode: Each node in the network automatically calculates a flooding topology by using the same algorithm (No flooding for flooding topology);

- Static Mode: Links on the flooding topology are configured statically.

Note that the flooding topology constructed by a node is dynamic in nature, that means when the base topology (the entire topology graph) changes, the flooding topology (the sub-graph) MUST be re-computed/ re-constructed to ensure that any node that is reachable on the base topology MUST also be reachable on the flooding topology.

For reference purpose, some algorithms that allow nodes to automatically compute flooding topology are elaborated in Appendix A.

However, this document does not attempt to standardize how a flooding topology is established.

5.2. Backup for Flooding Topology Split

It is hard to construct a flooding topology that reduces the amount of LS flooding greatly and is tolerant to multiple failures. To get around this, we can compute and use backup paths for a critical link and node on the flooding topology. Using backup paths may also speed up convergence when the link and node fail.

When a critical link on the flooding topology fails, the flooding topology without the critical link (i.e., the remaining flooding topology) is split into two parts. A backup path for the critical link connects the two parts into one. Through the backup path and the remaining flooding topology, an LS can be flooded to every node in the network. The combination of the backup path and the flooding topology is tolerant to the failure of the critical link.

When a critical node on the flooding topology goes down, the flooding topology without the critical node and the links attached to the node (i.e., the remaining flooding topology) is split into two or more parts. One or more backup paths for the critical node connects the split parts into one. Through the backup paths and the remaining flooding topology, an LS can be flooded to every live node in the network. The combination of the backup paths and the flooding topology is tolerant to the failure of the critical node.

In addition to the backup paths for a critical link and node, backup paths for every non critical link and node on the flooding topology can be computed. When the failures of multiple links and nodes on the flooding topology happen, through the remaining flooding topology and the backup paths for these links and nodes, an LS can be flooded to every live node in the network. The combination of the backup paths and the flooding topology is tolerant to the failures of these links and nodes. If there are other failures that break the backup paths, an LS can be flooded to every live node by the traditional flooding procedure.

In a centralized mode, the leader computes the backup paths and floods them to all the other nodes. In a distributed mode, every node computes the backup paths.

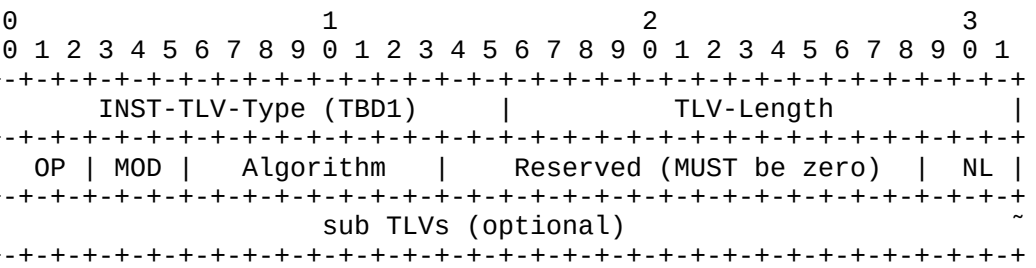
6. Extensions to OSPF

The extensions to OSPF comprises two parts: one part is for operations on flooding reduction, the other is specially for centralized mode flooding reduction.

6.1. Extensions for Operations

A new TLV is defined in OSPF RI LSA [RFC7770]. It contains instructions about flooding reduction, which is called Flooding Reduction Instruction TLV or Instruction TLV for short. This TLV is originated from only one node at any time.

The format of a Flooding Reduction Instruction TLV is as follows.



Flooding Reduction Instruction TLV

A OP field of three bits is defined in the TLV. It may have a value of the followings.

- 0x001 (R): Perform flooding Reduction, which instructs the nodes in a network to perform flooding reduction.

- 0x010 (N): Roll back to Normal flooding, which instructs the nodes in a network to roll back to perform normal flooding.

When any of the other values is received, it is ignored.

A MOD field of three bits is defined in the TLV and may have a value of the followings.

- 0x001 (C): Central Mode, which instructs 1) the nodes in a network to select leaders (primary/designated leader, secondary/backup leader, and so on); 2) the leaders in a network to compute a flooding topology and the primary leader to flood the flooding topology to all the other nodes in the network; 3) every node in the network to receive and use the flooding topology originated by the primary leader.

- 0x010 (D): Distributed Mode, which instructs every node in a network to compute and use its own flooding topology.

- 0x011 (S): Static Mode, which instructs every node in a network to use the flooding topology statically configured on the node.

When any of the other values is received, it is ignored.

An Algorithm field of eight bits is defined in the TLV to instruct the leader node in central mode or every node in distributed mode to use the algorithm indicated in this field for computing a flooding topology.

A NL field of three bits is defined in the TLV, which indicates the number of leaders to be selected when Central Mode is used. NL set to 2 means two leaders (a designated/primary leader and a backup/ secondary leader) to be selected for an area, and NL set to 3 means three leaders to be selected. When Central Mode is not used, The NL field is not valid.

Some optional sub TLVs may be defined in the future, but none is defined now.

6.2. Extensions for Centralized Mode

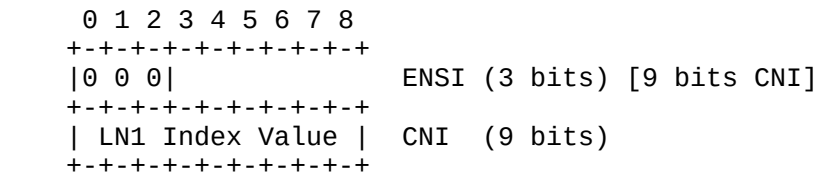
6.2.1. Message for Flooding Topology

A flooding topology can be represented by the links in the flooding topology. For the links between a local node and a number of its adjacent (or remote) nodes, we can encode the local node in a way, and encode its adjacent nodes in the same way or another way. After all the links in the flooding topology are encoded, the encoded links can be flooded to every node in the network. After receiving the encoded links, every node decodes the links and creates and/or updates the flooding topology.

For every node in an area, we may use an index to represent it. Every node in an area may order the nodes in a rule, which generates the same sequence of the nodes on every node in the area. The sequence of nodes have the index 0, 1, 2, and so on respectively. For example, every node orders the nodes by their router IDs in ascending order.

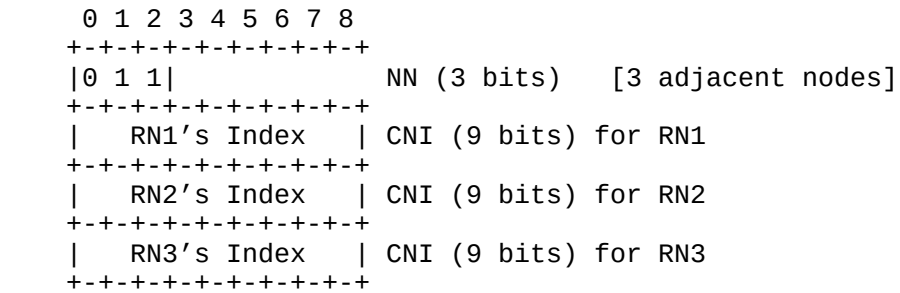
6.2.1.1. Links Encoding

A local node can be encoded in two parts: encoded node index size indication (ENSI) and compact node index (CNI). ENSI value plus a number (e.g., 9) gives the size of compact node index. For example, ENSI = 0 indicates that the size of CNIs is 9 bits. In the figure below, Local node LN1 is encoded as ENSI=0 using 3 bits and CNI=LN1’s Index using 9 bits. LN1 is encoded in 12 bits in total.



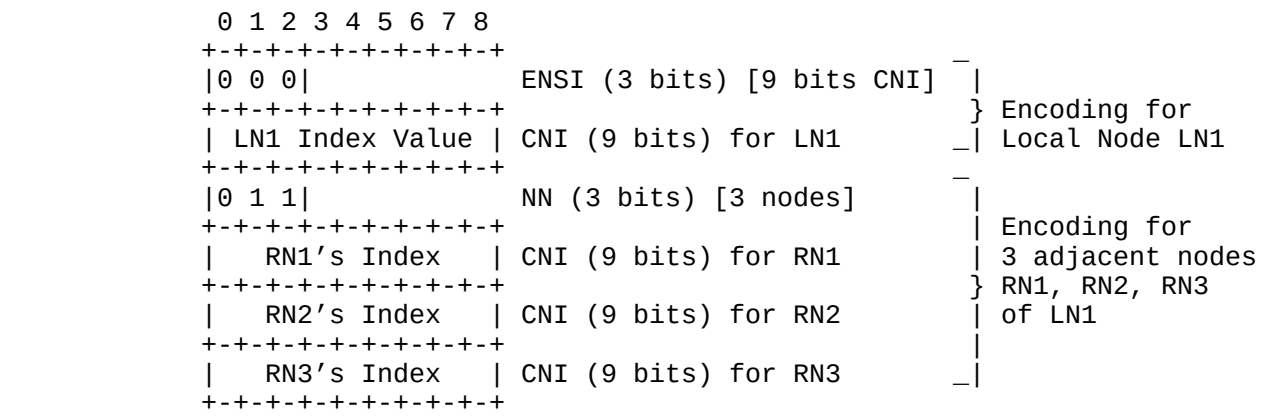
An Example of Local Node Encoding

The adjacent nodes can be encoded in two parts: Number of Nodes (NN) and compact node indexes (CNIs). The size of CNIs is the same as the local node. For example, three adjacent nodes RN1, RN2 and RN3 are encoded below in 30 bits (i.e., 3.75 bytes).



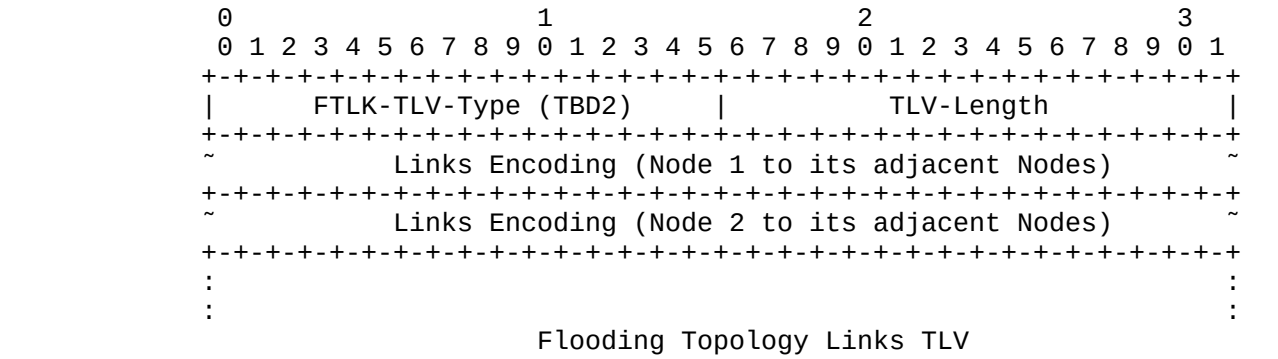
An Example of Adjacent Nodes Encoding

The links between a local node and a number of its adjacent (or remote) nodes can be encoded as the local node followed by the adjacent nodes. For example, three links between local node LN1 and its three adjacent nodes RN1, RN2 and RN3 are encoded below in 42 bits (i.e., 5.25 bytes).



An Example of Links Encoding

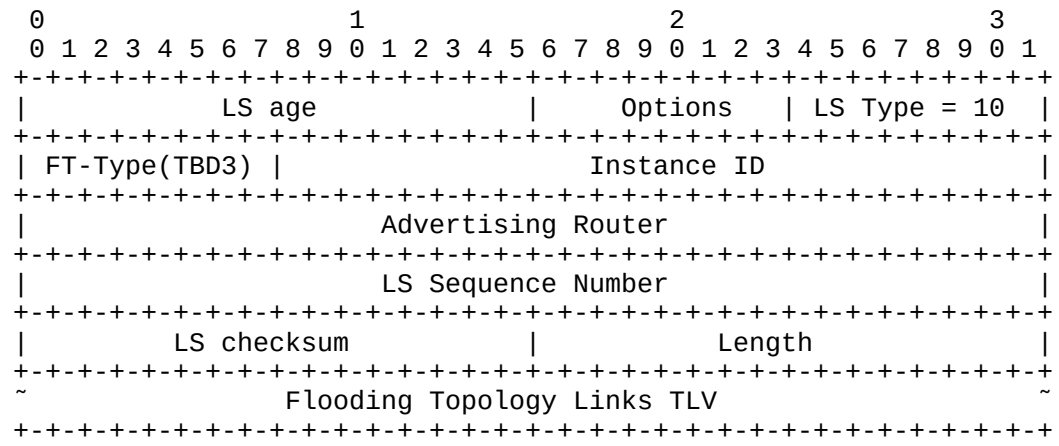
For a flooding topology computed by a leader of an area, it may be represented by all the links on the flooding topology. A Type-Length-Value (TLV) of the following format for the links encodings can be included in an LSA to represent the flooding topology (FT) and flood the FT to every node in the area.



Flooding Topology Links TLV

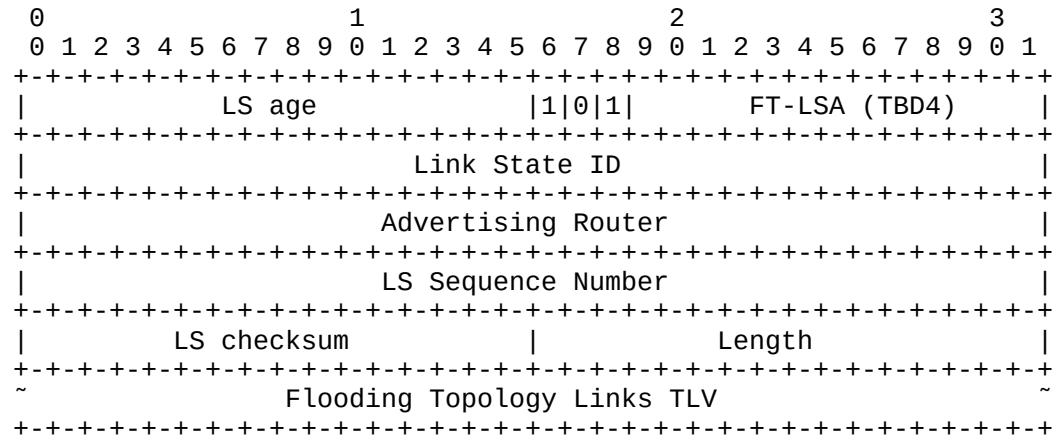
Note that a link between a local node LN and its adjacent node RN can be encoded once and as a bi-directional link. That is that if it is encoded in a Links Encoding from LN to RN, then the link from RN to LN is implied or assumed.

For OSPFv2, an Opaque LSA of a new opaque type (TBD3) containing a Flooding Topology Links TLV is used to flood the flooding topology from the leader of an area to all the other nodes in the area.



OSPFv2: Flooding Topology Opaque LSA

For OSPFv3, an area scope LSA of a new LSA function code (TBD4) containing a Flooding Topology Links TLV is used to flood the flooding topology from the leader of an area to all the other nodes in the area.



OSPFv3: Flooding Topology LSA

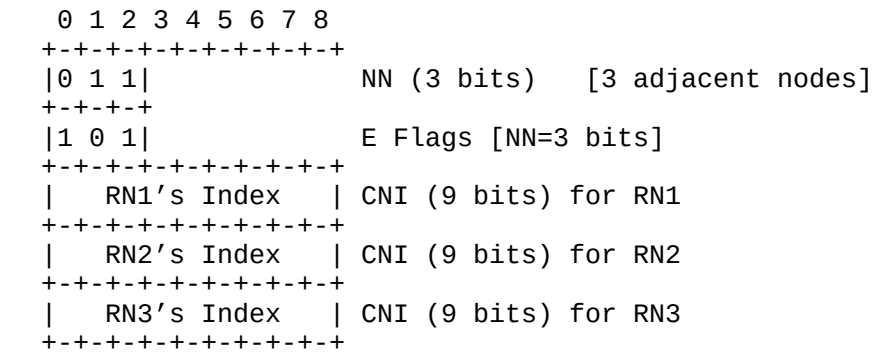
The U-bit is set to 1, and the scope is set to 01 for area-scoping.

6.2.1.2. Block Encoding

Block encoding uses a single structure to encode a block (or part) of topology, which can be a block of links in a flooding topology. It can also be all the links in the flooding topology. It starts with a local node LN and its adjacent (or remote) nodes RNi (i = 1, 2, ..., n), and can be considered as an extension to the links encoding.

The encoding of links between a local node and its adjacent nodes described in Section 6.2.1.1 is extended to include the links attached to the adjacent nodes.

The encoding for the adjacent nodes is extended to include Extending Flags (E Flags for short) between the NN (Number of Nodes) field and the CNIs (Compact Node Indexes) for the adjacent nodes. The length of the E Flags field is NN bits. The following is an example encoding of the adjacent nodes with E Flags of 3 bits, which is the value of the NN (the number of adjacent nodes).



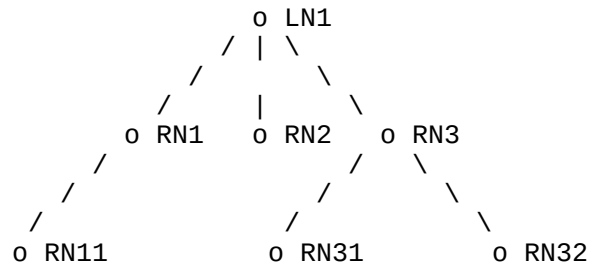
An Example of Adjacent Nodes with E Flags Encoding

There is a bit flag (called E flag) in the E Flags field for each adjacent node. The first bit (i.e., the most significant bit) in the E Flags field is for the first adjacent node (e.g., RN1), the second bit is for the second adjacent node (e.g., RN2), and so on. The E flag for an adjacent node RNi set to one indicates that the links attached to the adjacent node RNi are included below. The E flag for an adjacent node RNi set to zero means that no links attached to the adjacent node RNi are included below.

The links attached to the adjacent node RNi are represented by the RNi as a local node and the adjacent nodes of RNi. The encoding for the adjacent nodes of RNi is the same as that for the adjacent nodes

of a local node. It consists of an NN field of 3 bits, E Flags field of NN bits, and CNIs for the adjacent nodes of RNi.

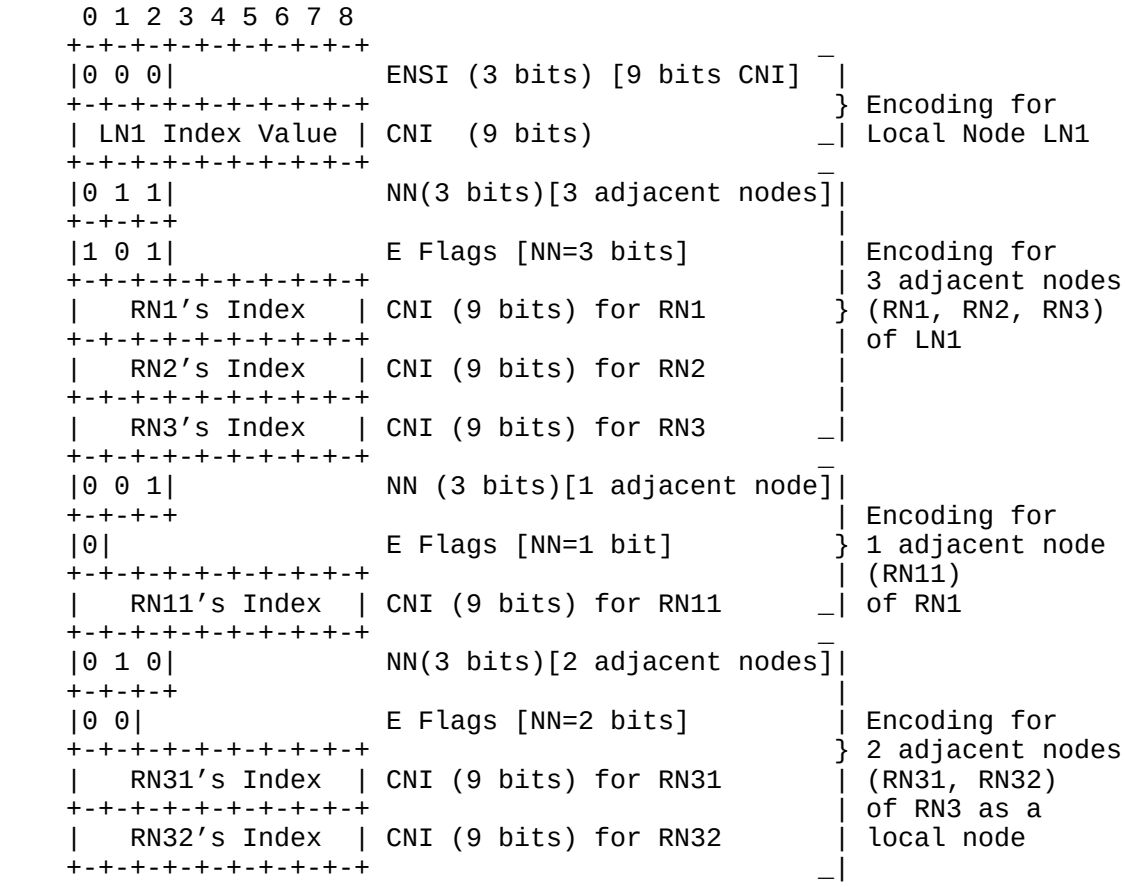
The following is an example of a block encoding for a block (or part) of flooding topology below.



An Example Block of Flooding Topology

It represents 6 links: 3 links between local node LN1 and its 3 adjacent nodes RN1, RN2 and RN3; 1 link between RN1 as a local node and its 1 adjacent node RN11; and 2 links between RN3 as a local node and its 2 adjacent nodes RN31 and RN32.

It starts with the encoding of the links between local node LN1 and 3 adjacent nodes RN1, RN2 and RN3 of the local node LN1. The encoding for the local node LN1 is the same as that for a local node described in Section 6.2.1.1. The encoding for 3 adjacent nodes RN1, RN2 and RN3 of local node LN1 comprises an NN field of 3 bits with value of 3, E Flags field of NN = 3 bits, and the indexes of adjacent nodes RN1, RN2 and RN3.



An Example of Block Encoding

The first E flag in the encoding for adjacent nodes RN1, RN2 and RN3 is set to one, which indicates that the links between the first adjacent node RN1 as a local node and its adjacent nodes are included below. In this example, 1 link between RN1 and its adjacent node RN11 is represented by the encoding for the adjacent node RN11 of RN1 as a local node. The encoding for 1 adjacent node RN11 consists of an NN field of 3 bits with value of 1, E Flags field of NN = 1 bits, and the index of adjacent node RN11. The size of the index of RN11 is the same as that of local node LN1 indicated by the ENSI in the encoding for local node LN1.

The second E flag in the encoding for adjacent nodes RN1, RN2 and RN3 is set to zero, which indicates that no links between the second

adjacent node RN2 as a local node and its adjacent nodes are included below.

The third E flag in the encoding for adjacent nodes RN1, RN2 and RN3 is set to one, which indicates that the links between the third adjacent node RN3 as a local node and its adjacent nodes are included below. In this example, 2 links between RN3 and its 2 adjacent nodes RN31 and RN32 are represented by the encoding for the adjacent nodes RN31 and RN32 of RN3 as a local node. The encoding for 2 adjacent nodes RN31 and RN32 consists of an NN field of 3 bits with value of 2, E Flags field of NN = 2 bits, and the indexes of adjacent nodes RN31 and RN32. The size of the index of RN31 and RN32 is the same as that of local node LN1 indicated by the ENSI in the encoding for local node LN1.

The block encoding may be used in the place of the links encoding in Section 6.2.1.1 for more efficiency. That is that it may be used in a Flooding Topology Links TLV. Alternatively, a new TLV, which is similar to the Flooding Topology Links TLV, may be defined to contain a number of block encodings.

6.2.2. Encodings for Backup Paths

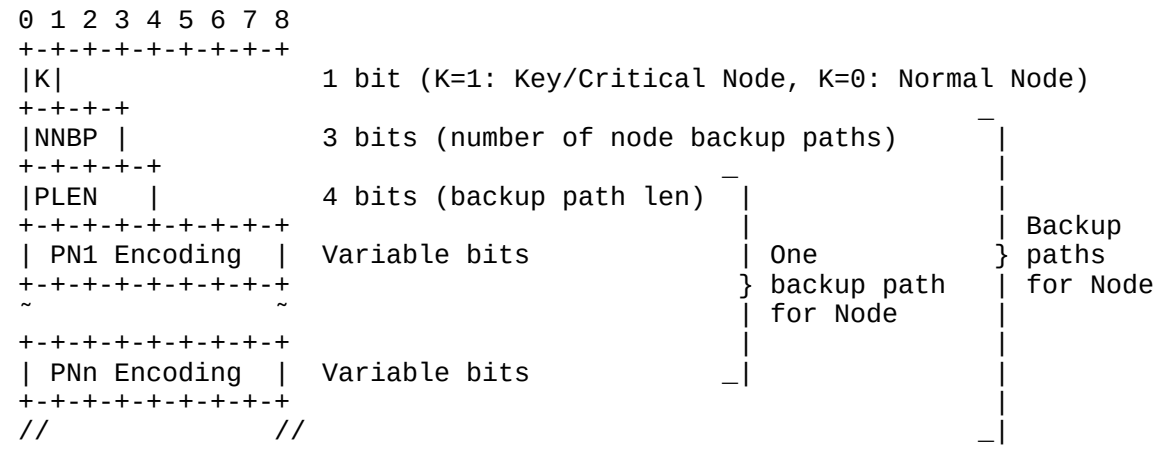
When the leader of an area computes a flooding topology, it may compute a backup path or multiple backup paths for a critical link on the flooding topology. When the critical link fails, a link state can be distributed to every node in the area through one backup path and other links on the flooding topology. In addition, it may compute a backup path or multiple backup paths for a node. When the node fails, a link state can be distributed to the other nodes in the area through the backup paths and the links on the flooding topology.

This section describes two encodings for backup paths: separated encoding and integrated one. In the former, backup paths are encoded in a new message, where the message for the flooding topology described in the previous section is required; In the latter, backup paths are integrated into the flooding topology links encoding, where one message contains the flooding topology and the backup paths.

6.2.2.1. Message for Backup Paths

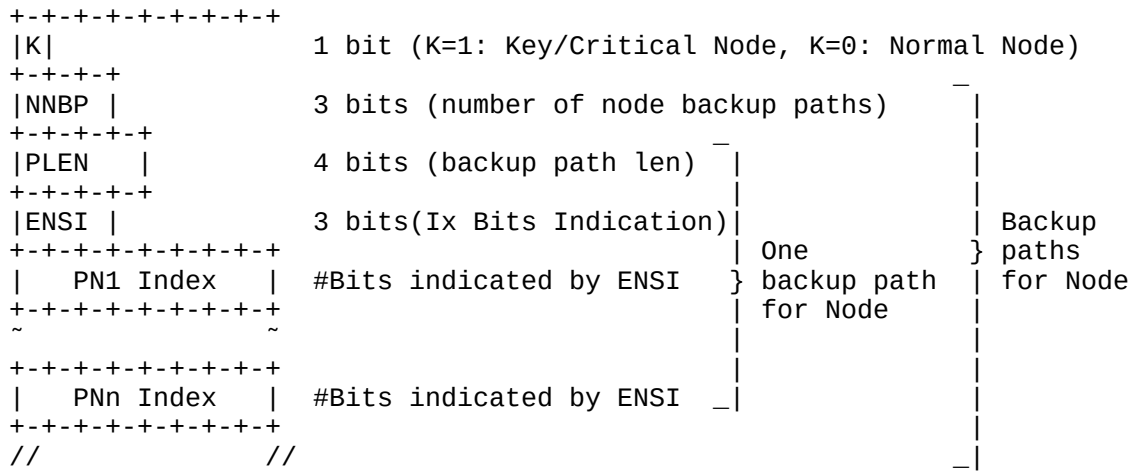
Backup paths for a node (such as Node1) may be represented by the node index encoding and node backup paths encoding. The former is similar to local node index encoding. The latter has the following format. It comprises a K flag (Key/Critical node flag) of 1 bit, a 3 bits NNBP field (number of node backup paths), and each of the backup paths encoding, which consists of the path length PLEN of 4 bits indicating the length of the path (i.e., the number of nodes), and

the encoding of the sequence of nodes along the path such as encodings for nodes PN1, ..., PNn. The encoding of every node may use the encoding of a local node, which comprises encoded node index size indication (ENSI) and compact node index (CNI).



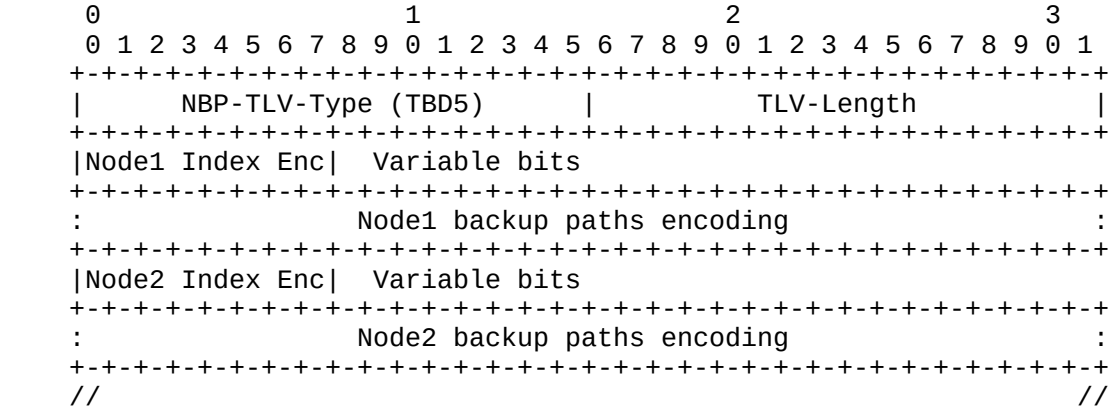
An Example of Node Backup Paths Encoding

Another encoding of the sequence of nodes along the path uses one encoded node index size indication (ENSI) for all the nodes in the path. Thus we have the following Node Backup Paths Encoding.



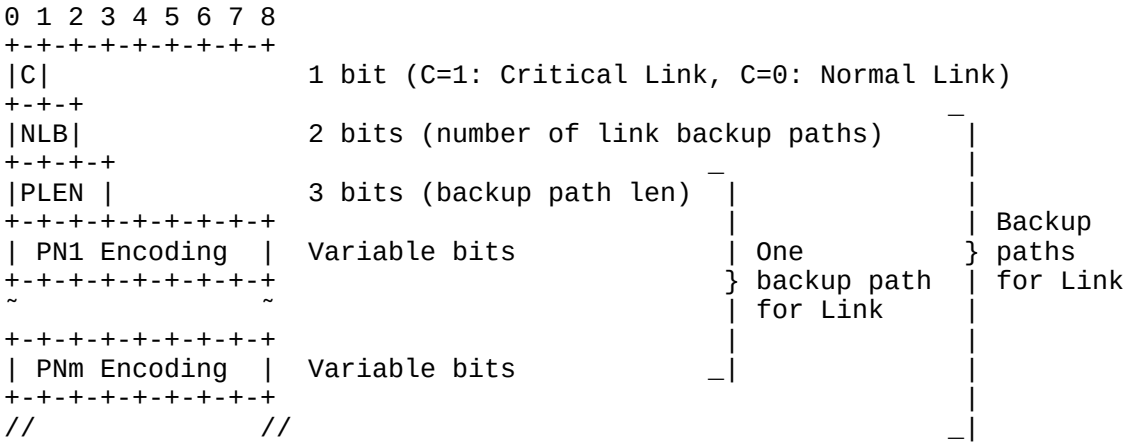
Another Example of Node Backup Paths Encoding

A new TLV called Node Backup Paths TLV is defined below. It may include multiple nodes and their backup paths. Each node is represented by its index encoding, which is followed by its node backup paths encoding.



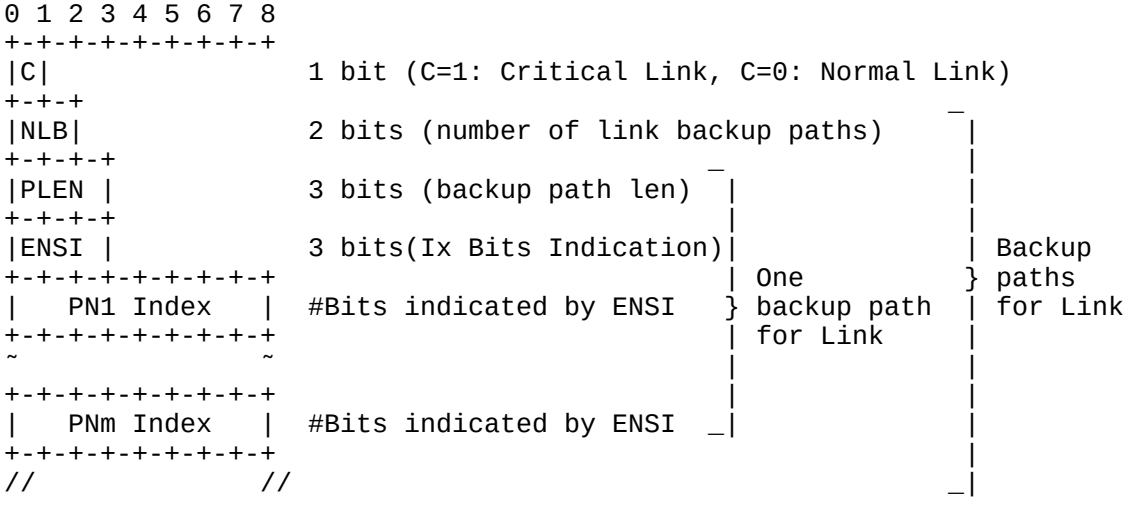
Node Backup Paths TLV

The encoding for backup paths for a link (such as Link1) on the flooding topology consists of the link encoding such as Link1 Index Encoding and the link backup paths encoding. The former is similar to local node encoding. It contains encoded link index size indication (ELSI) and compact link index (CLI). The latter has the following format. It comprises a C flag (Critical link flag) of 1 bit, a 2 bits NLB field (number of link backup paths), and each of the backup paths encoding, which consists of the path length PLEN of 3 bits indicating the length of the path (i.e., the number of nodes), and the encoding of the sequence of nodes along the path such as encodings for nodes PN1, ..., PNm. Note that two ends of a link (i.e., the local node and the adjacent/remote node of the link) are not needed in the path. The encoding of every node may use the encoding of a local node, which comprises encoded node index size indication (ENSI) and compact node index (CNI).



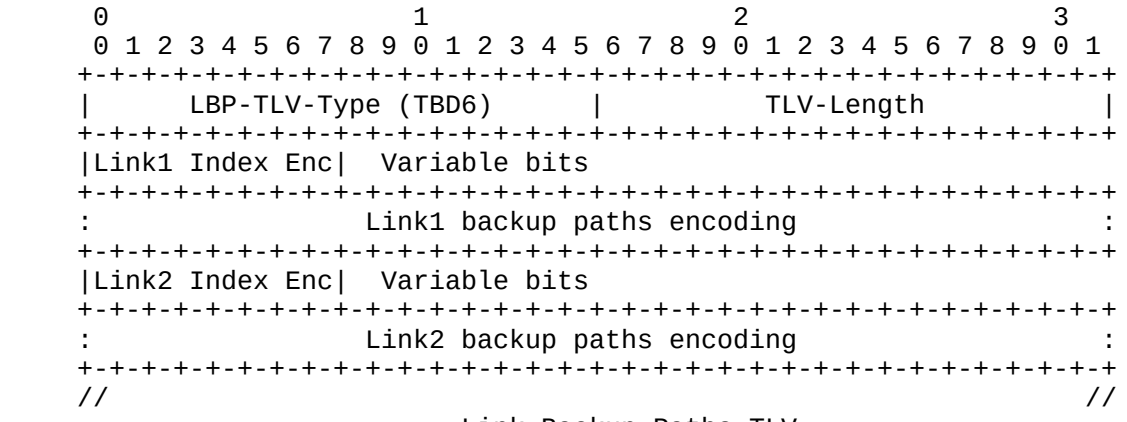
An Example of Link Backup Paths Encoding

Another encoding of the sequence of nodes along the path uses one encoded node index size indication (ENSI) for all the nodes in the path. Thus we have the following Link Backup Paths Encoding.



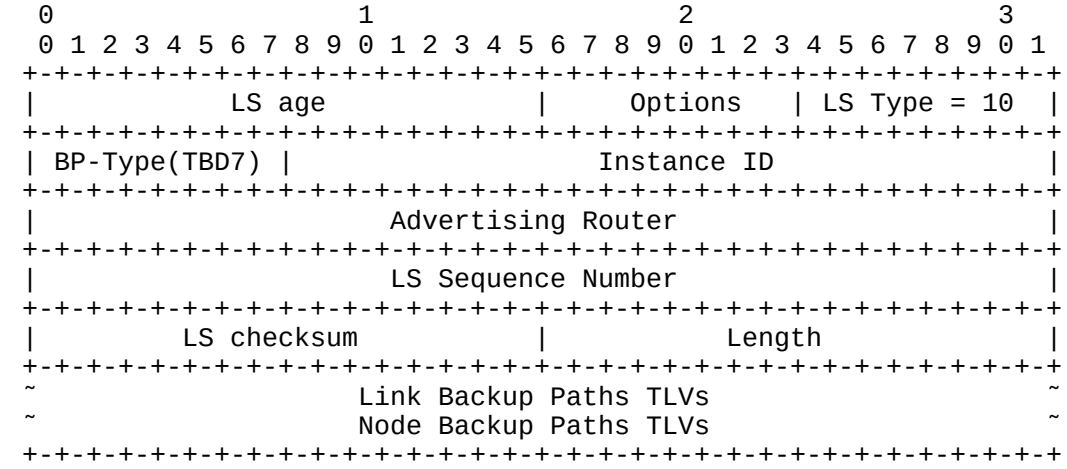
Another Example of Link Backup Paths Encoding

A new TLV called Link Backup Paths TLV is defined below. It may include multiple links and their backup paths. Each link is represented by its index encoding, which is followed by its link backup paths encoding.



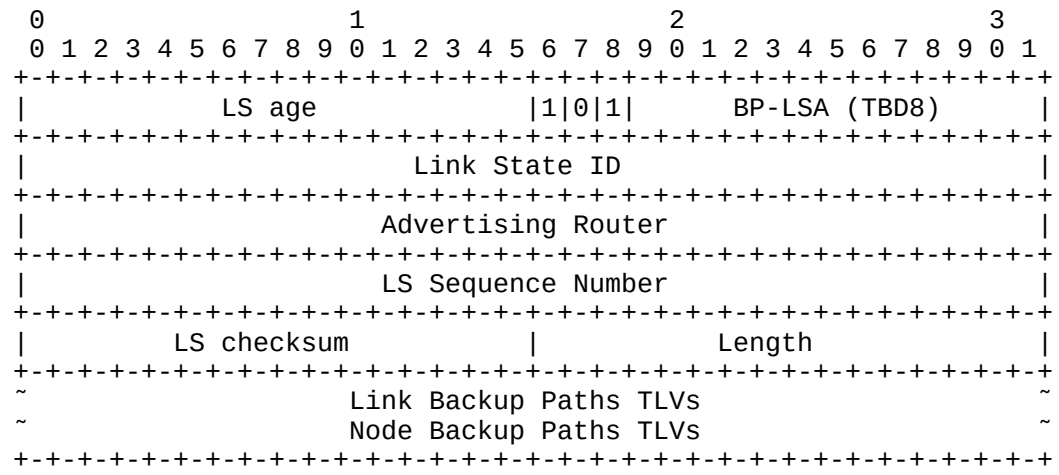
Link Backup Paths TLV

For OSPFv2, an Opaque LSA of a new opaque type (TBD7), containing node backup paths TLVs and link backup paths TLVs, is used to flood the backup paths from the leader of an area to all the other nodes in the area.



OSPFv2: Backup Paths Opaque LSA

For OSPFv3, an area scope LSA of a new LSA function code (TBD8), containing node backup paths TLVs and link backup paths TLVs, is used to flood the backup paths from the leader of an area to all the other nodes in the area.

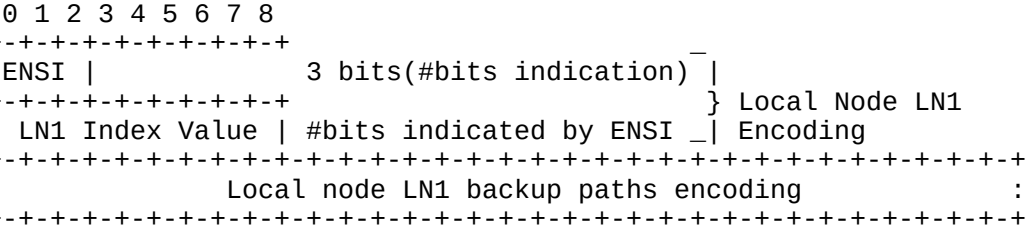


OSPFv3: Backup Paths LSA

The U-bit is set to 1, and the scope is set to 01 for area-scoping.

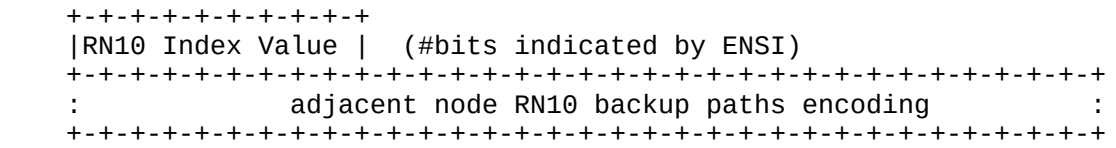
6.2.2.2. Backup Paths in Links TLV

A local node and its backup paths can be encoded in the following format. It is the local node (such as local node LN1) encoding followed by the local node backup paths encoding, which is the same as the node backup paths encoding described in Section 6.2.2.1.



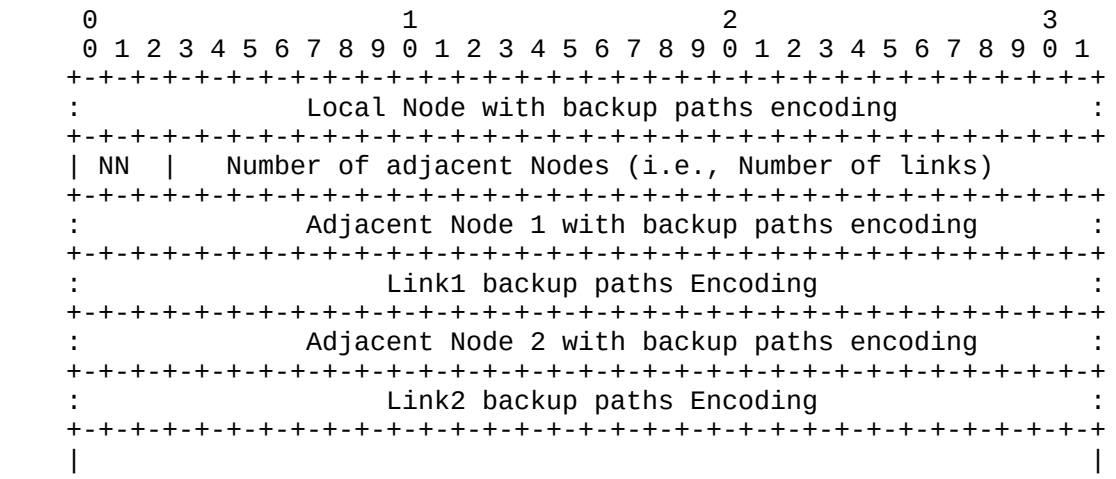
Local Node with Backup Paths Encoding

A adjacent node and its backup paths can be encoded in the following format. It is the adjacent node (such as adjacent node RN10) index value followed by the adjacent node backup paths encoding, which is the same as the node backup paths encoding described in Section 6.2.2.1.



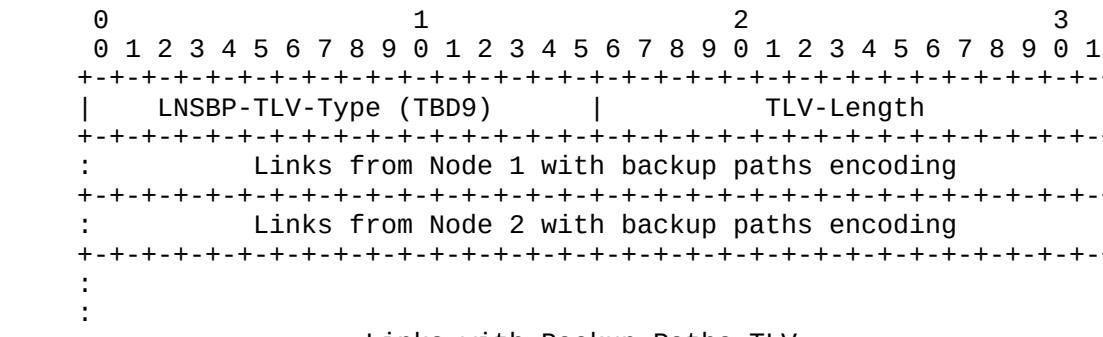
Adjacent Node with Backup Paths Encoding

The links between a local node and a number of its adjacent nodes, the backup paths for each of the nodes, and the backup paths for each of the links can be encoded in the following format. It is called Links from Node with Backup Paths Encoding.



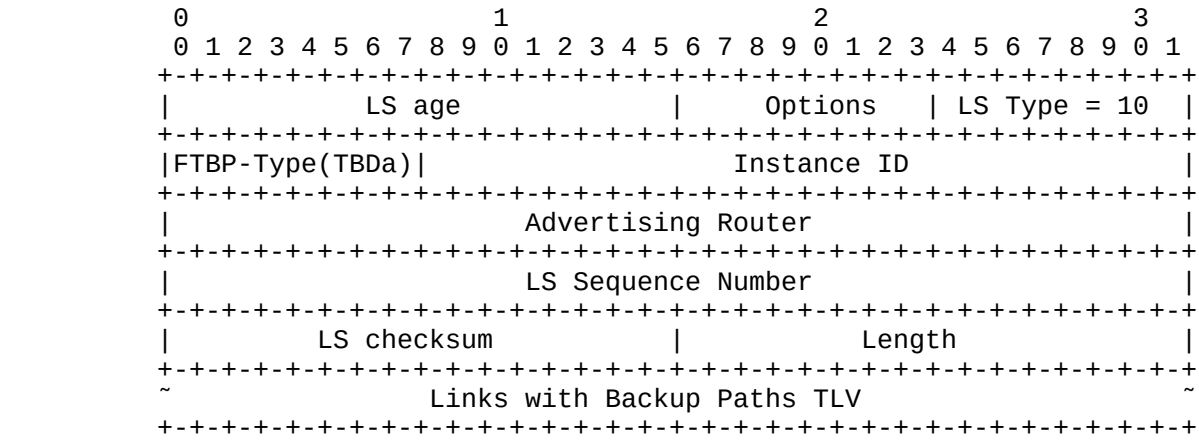
Links from Node with Backup Paths Encoding

A new TLV called Links with Backup Paths TLV is defined below. It includes a number of Links from Node with Backup Paths Encodings described above. This TLV contains both the flooding topology and the backup paths for the links and nodes on the flooding topology.



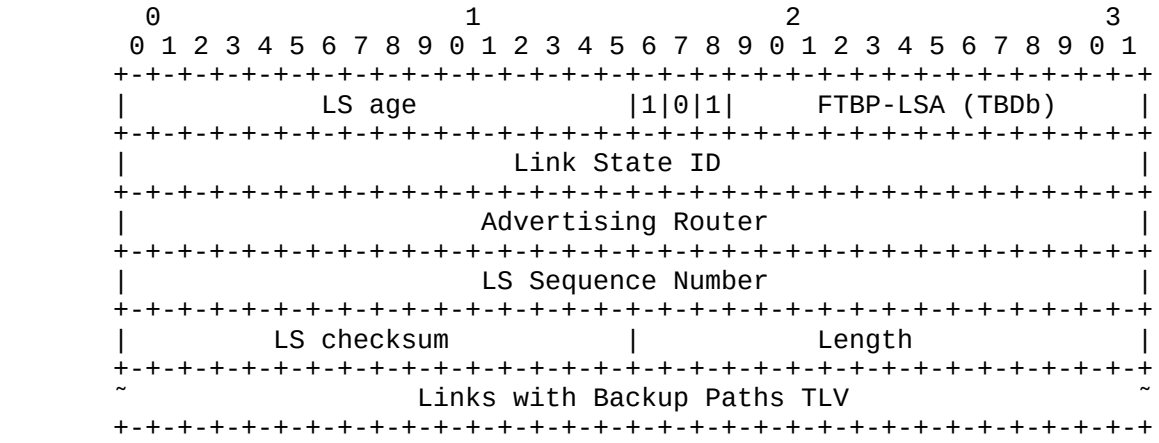
Links with Backup Paths TLV

For OSPFv2, an Opaque LSA of a new opaque type (TBDa), called Flooding Topology with Backup Paths (FTBP) Opaque LSA, containing a Links with Backup Paths TLV, is used to flood the flooding topology with backup paths from the leader of an area to all the other nodes in the area.



OSPFv2: Flooding Topology with Backup Paths (FTBP) Opaque LSA

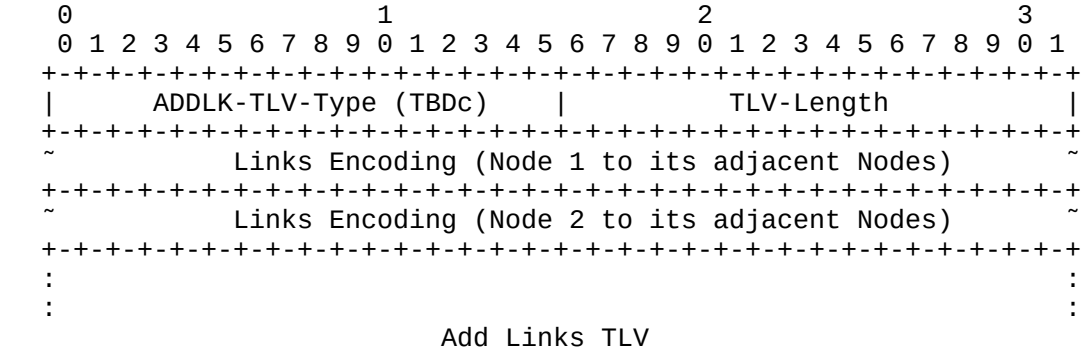
For OSPFv3, an area scope LSA of a new LSA function code (TBDb), containing a Links with Backup Paths TLV, is used to flood the flooding topology with backup paths from the leader of an area to all the other nodes in the area.



OSPFv3: Flooding Topology with Backup Paths (FTBP) LSA

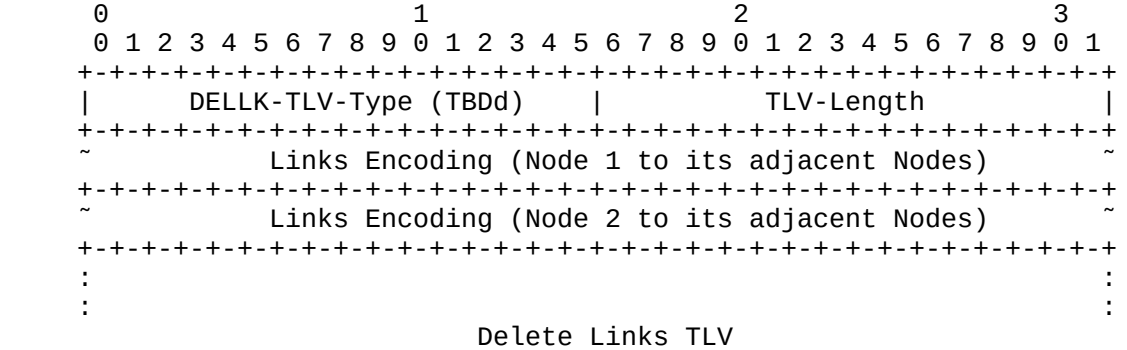
6.2.3. Message for Incremental Changes

For adding some links to the flooding topology, we define a new TLV called Add Links TLVs of the following format. When some new links are added to the flooding topology, the leader may not flood the whole flooding topology with the new links to all the other nodes. It may just flood these new links. After receiving these new links, each of the other nodes adds these new links into the existing flooding topology. When the leader floods the whole flooding topology with the new links to all the other nodes, it removes the LSA for the new links. When removing the LSA for these new links, each of the other nodes does not update the flooding topology (i.e., does not remove these links from the flooding topology).



Add Links TLV

For deleting some links from the flooding topology, we define a new TLV called Delete Links TLVs of the following format. When some old links are removed from the flooding topology, the leader may not flood the whole flooding topology without the old links to all the other nodes. It may just flood these old links. After receiving these old links, each of the other nodes deletes these old links from the existing flooding topology. When the leader floods the whole flooding topology without the old links to all the other nodes, it removes the LSA for the old links. When removing the LSA for these old links, each of the other nodes does not update the flooding topology (i.e., does not add these links into the flooding topology).



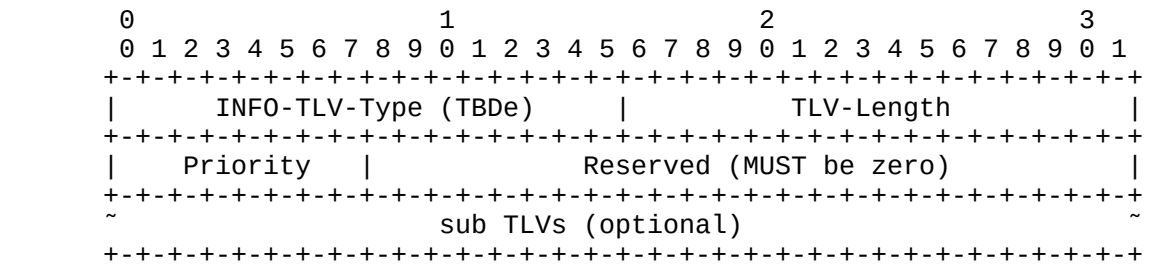
Delete Links TLV

The Add Links TLVs and Delete Links TLVs should be in a separate LSA instance. The LSA can be a Flooding Topology LSA defined above. Alternatively, we may define a new LSA for these TLVs.

6.2.4. Leaders Selection

The leader or Designated Router (DR) selection for a broadcast link is about selecting two leaders: a DR and Backup DR. This is generalized to select two or more leaders for an area: the primary/ first leader (or leader for short), the secondary leader, the third leader and so on.

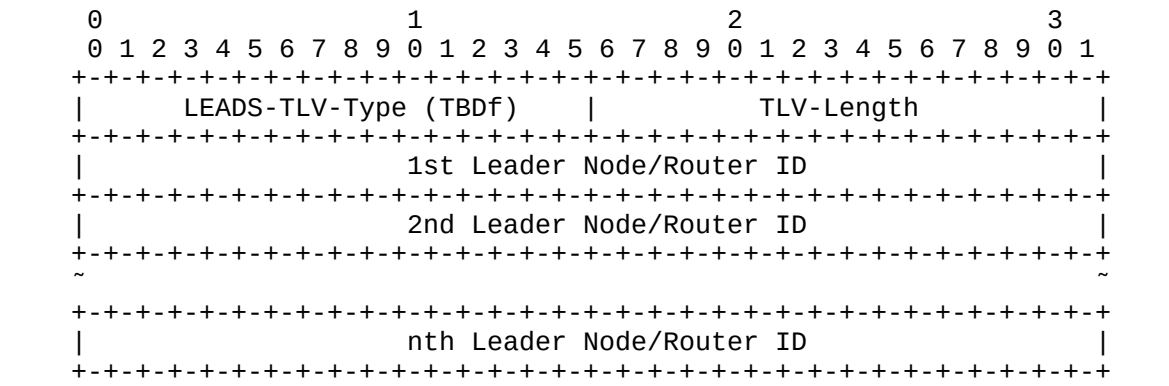
A new TLV is defined to include the information on flooding reduction of a node, which is called Flooding Reduction Information TLV or Information TLV for short. This TLV is generated by every node that supports flooding reduction in general. Every node originates a RI LSA with a Flooding Reduction Information TLV containing its priority to become a leader. The format of the TLV is as follows.



Flooding Reduction Information TLV

A Priority field of eight bits is defined in the TLV to indicate the priority of the node originating the TLV to become the leader node in central mode.

A sub-TLV called leaders sub-TLV is defined. It has the following format.



Leaders sub-TLV

When a node selects itself as a leader, it originates a RI LSA containing the leader in a leaders sub-TLV.

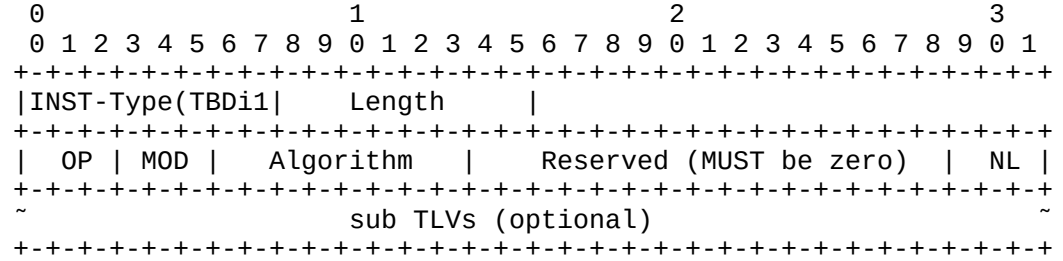
After the first leader node is down, the other leaders will be promoted. The secondary leader becomes the first leader, the third leader becomes the secondary leader, and so on. When a node selects itself as the n-th leader, it originates a RI LSA with a Leaders sub-TLV containing n leaders.

7. Extensions to IS-IS

The extensions to IS-IS is similar to OSPF.

7.1. Extensions for Operations

A new TLV for operations is defined in IS-IS LSP. It has the following format and contains the same contents as the Flooding Reduction Instruction TLV defined in OSPF RI LSA.

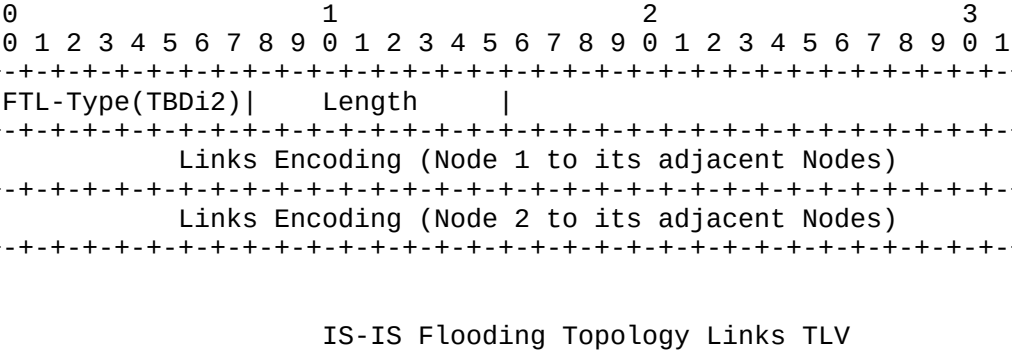


IS-IS Flooding Reduction Instruction TLV

7.2. Extensions for Centralized Mode

7.2.1. TLV for Flooding Topology

A new TLV for the encodings of the links in the flooding topology is defined. It has the following format and contains the same contents as the Flooding Topology Links TLV defined in OSPF Flooding Topology Opaque LSA.

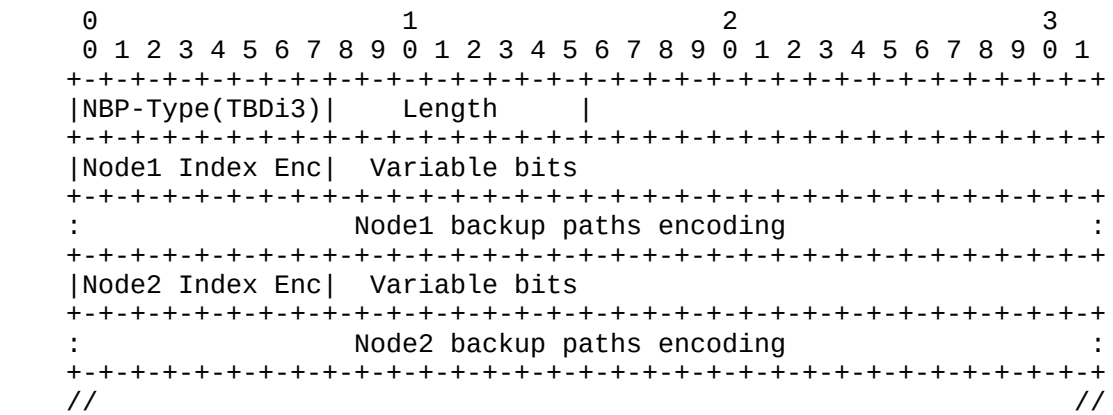


IS-IS Flooding Topology Links TLV

7.2.2. Encodings for Backup Paths

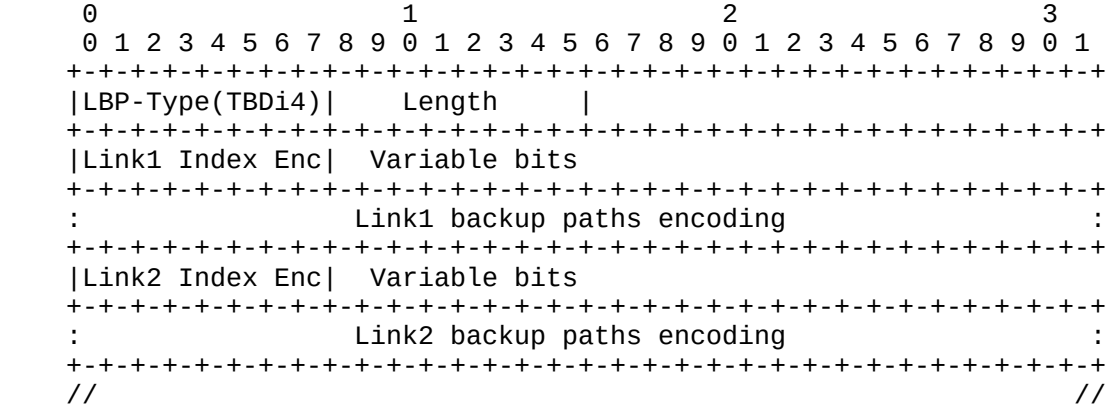
7.2.2.1. TLVs for Backup Paths

For flooding backup paths separately, we define two TLVs: IS-IS Node Backup Paths TLV and IS-IS Link Backup Path TLV. The former has the following format and contains the same contents as Node Backup Paths TLV in OSPF.



IS-IS Node Backup Paths TLV

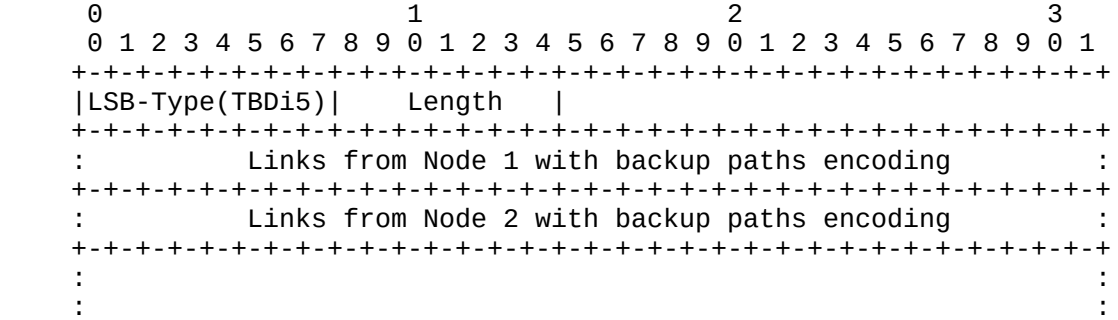
The latter has the following format and contains the same contents as Link Backup Paths TLV in OSPF.



IS-IS Link Backup Paths TLV

7.2.2.2. Backup Paths in Links TLV

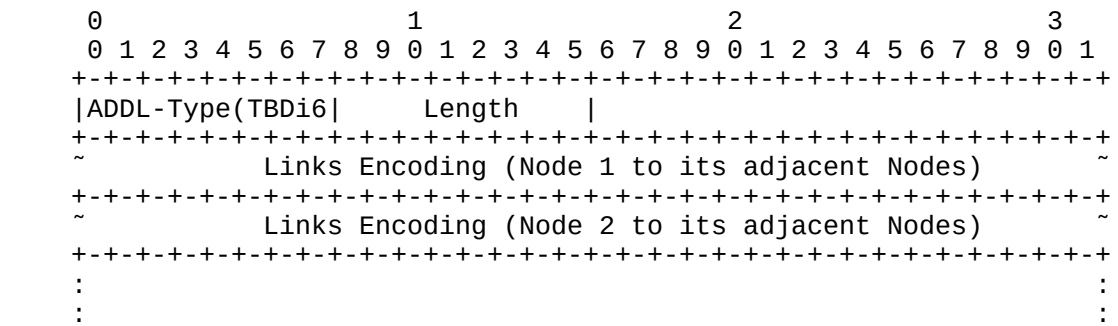
A new TLV is defined to integrate the backup paths with the links on the flooding topology. It has the following format and contains the same contents as the Links with Backup Paths TLV in OSPF.



IS-IS Links with Backup Paths TLV

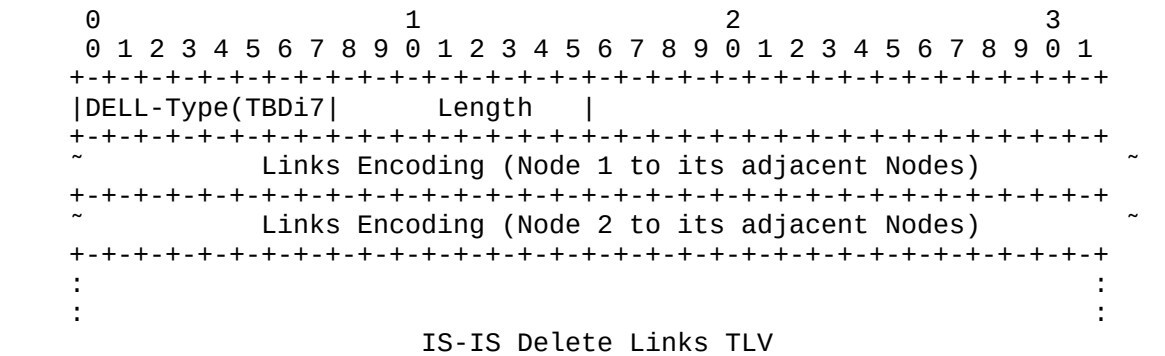
7.2.3. TLVs for Incremental Changes

Similar to Add Links TLV in OSPF, a new TLV called IS-IS Add Links TLV is defined. It has the following format and contains the same contents as Add Links TLV in OSPF.



IS-IS Add Links TLV

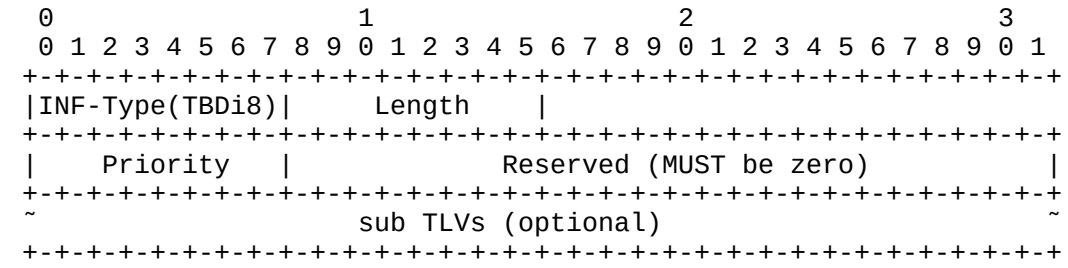
Similar to Delete Links TLV in OSPF, a new TLV called IS-IS Delete Links TLV is defined. It has the following format and contains the same contents as Delete Links TLV in OSPF.



IS-IS Delete Links TLV

7.2.4. Leaders Selection

Similar to Flooding Reduction Information TLV in OSPF, a new TLV called IS-IS Flooding Reduction Information TLV is defined. It has the following format and contains the same contents as Flooding Reduction Information TLV in OSPF.



IS-IS Flooding Reduction Information TLV

8. Flooding Behavior

This section describes the revised flooding behavior for a node having at least one link on the flooding topology. The revised flooding procedure MUST flood an LS to every node in the network in any case, as the standard flooding procedure does.

8.1. Nodes Perform Flooding Reduction without Failure

8.1.1. Receiving an LS

When a node receives a newer LS that is not originated by itself from one of its interfaces, it floods the LS only to all the other interfaces that are on the flooding topology.

When the LS is received from an interface on the flooding topology, it is flooded only to all the other interfaces that are on the flooding topology. When the LS is received on an interface that is not on the flooding topology, it is also flooded only to all the other interfaces that are on the flooding topology.

In any case, the LS must not be transmitted back to the receiving interface.

Note before forwarding a received LS, the node would do the normal processing as usual.

8.1.2. Originating an LS

When a node originates an LS, it floods the LS to its interfaces on the flooding topology if the LS is a refresh LS (i.e., there is no significant change in the LS comparing to the previous LS); otherwise (i.e., there are significant changes in the LS), it floods the LS to all its interfaces. Choosing flooding the LS with significant changes to all the interfaces instead of limiting to the interfaces on the flooding topology would speed up the distribution of the significant link state changes.

8.1.3. Establishing Adjacencies

Adjacencies being established can be classified into two categories: adjacencies to new nodes and adjacencies to existing nodes.

8.1.3.1. Adjacency to New Node

An adjacency to a new node is an adjacency between a node (say node A) on the flooding topology and the new node (say node Y) which is not on the flooding topology. There is not any adjacency between node Y and a node in the network area.

When new node Y is up and connected to node A, node A assumes that node Y and the link between node Y and node A are on the flooding topology until a new flooding topology is computed and built. Node A may determine whether node Y is a new node through checking if node Y is reachable or on the flooding topology.

The procedure for establishing the adjacency between node A and node Y is the existing normal procedure unchanged. After the status of the adjacency reaches to Exchange or Full, node A sends node Y every new or updated LS that node A receives or originates.

8.1.3.2. Adjacency to Existing Node

An adjacency to an existing node is an adjacency between a node (say node A) on the flooding topology and the existing node (say node X) which exists on the flooding topology. There are some adjacencies between node X and some nodes in the network area.

When existing node X is connected to node A after a link between node X and node A is up, node A assumes that the link connecting node A and node X is not on the flooding topology until a new flooding topology is computed and built. Node A may determine whether node X is an existing node through checking if node X is reachable or on the flooding topology.

The procedure for establishing the adjacency between node A and node X is the existing normal procedure unchanged. Node A does not send node X any new or updated LS that node A receives or originates even after the status of the adjacency reaches to Exchange or Full.

8.2. An Exception Case

During an LS flooding, one or multiple link and node failures may happen. Some failures do not split the flooding topology, thus do not affect the flooding behavior. For example, multiple failures of the links not on the flooding topology do not split the flooding topology and do not affect the flooding behavior. The sections below focus on the failures that may split the flooding topology.

8.2.1. A Critical Failure

For a link failure, if the link is a critical link on the flooding topology, then the LS is flooded through a backup path for the link and the remaining flooding topology until a new flooding topology is computed and built; otherwise, the flooding behavior in Section 8.1 follows.

Similarly, for a node failure, if the node is a critical node on the flooding topology, then the LS is flooded through backup paths for the node and the remaining flooding topology until a new flooding topology is computed and built; otherwise, the flooding behavior in Section 8.1 follows.

8.2.2. Multiple Failures

For multiple link failures, if the number of the failed links on the flooding topology is greater than or equal to two, then the LS is flooded through a backup path for each of the failed links on the flooding topology and the remaining flooding topology until a new

flooding topology is computed and built; otherwise, the flooding behavior in Section 8.1 follows.

If all the backup paths for some of the failed links are broken by some failures, the LS is flooded to all interfaces (except where it is received from) until a new flooding topology is computed and built.

For multiple node failures, the LS is flooded through the backup paths for each of the failed nodes and the remaining flooding topology until a new flooding topology is computed and built; otherwise, the flooding behavior in Section 8.1 follows.

If the backup paths for some of the failed nodes are broken by some failures, the LS is flooded to all interfaces (except where it is received from) until a new flooding topology is computed and built.

Note that if it can be quickly determined that the flooding topology is not split by the failures, the flooding behavior in Section 8.1 may follow.

9. Security Considerations

This document does not introduce any security issue.

10. IANA Considerations

10.1. OSPFv2

Under Registry Name: OSPF Router Information (RI) TLVs [RFC7770], IANA is requested to assign two new TLV values for OSPF flooding reduction as follows:

TLV Value	TLV Name	reference
11	Instruction TLV	This document
12	Information TLV	This document

Under the registry name "Opaque Link-State Advertisements (LSA) Option Types" [RFC5250], IANA is requested to assign new Opaque Type registry values for FT LSA, BP LSA, FTBP LSA as follows:

Registry Value	Opaque Type	reference
10	FT LSA	This document
11	BP LSA	This document
12	FTBP LSA	This document

IANA is requested to create and maintain new registries:

o OSPFv2 FT LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv2 FT LSA TLV Name	Definition
0	Reserved	
1	FT Links TLV	see Section 6.2.1
2-32767	Unassigned	
32768-65535	Reserved	

o OSPFv2 BP LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv2 TBPLSA TLV Name	Definition
0	Reserved	
1	Node Backup Paths TLV	see Section 6.2.2
2	Link Backup Paths TLV	see Section 6.2.2
3-32767	Unassigned	
32768-65535	Reserved	

o OSPFv2 FTBP LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv2 FTBP LSA TLV Name	Definition
0	Reserved	
1	Links with Backup Paths TLV	see Section 6.2.2
2-32767	Unassigned	
32768-65535	Reserved	

10.2. OSPFv3

Under the registry name "OSPFv3 LSA Function Codes", IANA is requested to assign new registry values for FT LSA, BP LSA, FTBP LSA as follows:

Value	LSA Function Code Name	reference
16	FT LSA	This document
17	BP LSA	This document
18	FTBP LSA	This document

IANA is requested to create and maintain new registries:

- OSPFv3 FT LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv3 FT LSA TLV Name	Definition
0	Reserved	
1	FT Links TLV	see Section 6.2.1
2-32767	Unassigned	
32768-65535	Reserved	

- OSPFv3 BP LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv3 TBPLSA TLV Name	Definition
0	Reserved	
1	Node Backup Paths TLV	see Section 6.2.2
2	Link Backup Paths TLV	see Section 6.2.2
3-32767	Unassigned	
32768-65535	Reserved	

- OSPFv3 FTBP LSA TLVs

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	OSPFv3 FTBP LSA TLV Name	Definition
0	Reserved	
1	Links with Backup Paths TLV	see Section 6.2.2
2-32767	Unassigned	
32768-65535	Reserved	

10.3. IS-IS

Under Registry Name: IS-IS TLV Codepoints, IANA is requested to assign new TLV values for IS-IS flooding reduction as follows:

Value	TLV Name	Definition
151	FT Links TLV	see Section 7.2.1
152	Node Backup Paths TLV	see Section 7.2.2
153	Link Backup Paths TLV	see Section 7.2.2
154	Links with Backup Paths TLV	see Section 7.2.2
155	Add Links TLV	see Section 7.2.3
156	Delete Links TLV	see Section 7.2.3
157	Instruction TLV	see Section 7.1
158	Information TLV	see Section 7.2.4

11. Acknowledgements

The authors would like to thank Acee Lindem, Zhibo Hu, Robin Li, Stephane Litkowski and Alvaro Retana for their valuable suggestions and comments on this draft.

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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, DOI 10.17487/RFC5250, July 2008, <https://www.rfc-editor.org/info/rfc5250>.

[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>.

[RFC7770] Lindem, A., Ed., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 7770, DOI 10.17487/RFC7770, February 2016, <https://www.rfc-editor.org/info/rfc7770>.

12.2. Informative References

[I-D.li-dynamic-flooding]
Li, T. and P. Psenak, "Dynamic Flooding on Dense Graphs", draft-li-dynamic-flooding-05 (work in progress), June 2018.

[I-D.shen-isis-spine-leaf-ext]
Shen, N., Ginsberg, L., and S. Thyamagundalu, "IS-IS Routing for Spine-Leaf Topology", draft-shen-isis-spine-leaf-ext-06 (work in progress), June 2018.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

Appendix A. Algorithms to Build Flooding Topology

There are many algorithms to build a flooding topology. A simple and efficient one is briefed below.

- Select a node R according to a rule such as the node with the biggest/smallest node ID;
- Build a tree using R as root of the tree (details below); and then
- Connect k (k>=0) leaves to the tree to have a flooding topology (details follow).

A.1. Algorithms to Build Tree without Considering Others

An algorithm for building a tree from node R as root starts with a candidate queue Cq containing R and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft
2. If Cq is empty, then return with Ft

3. Suppose that node Xi (i = 1, 2, ..., n) is connected to node A and not in Ft and X1, X2, ..., Xn are in a special order. For example, X1, X2, ..., Xn are ordered by the cost of the link between A and Xi. The cost of the link between A and Xi is less than the cost of the link between A and Xj (j = i + 1). If two costs are the same, Xi's ID is less than Xj's ID. In another example, X1, X2, ..., Xn are ordered by their IDs. If they are not ordered, then make them in the order.

4. Add Xi (i = 1, 2, ..., n) into the end of Cq, goto step 1.

Another algorithm for building a tree from node R as root starts with a candidate queue Cq containing R and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft

2. If Cq is empty, then return with Ft

3. Suppose that node Xi (i = 1, 2, ..., n) is connected to node A and not in Ft and X1, X2, ..., Xn are in a special order. For example, X1, X2, ..., Xn are ordered by the cost of the link between A and Xi. The cost of the link between A and Xi is less than the cost of the link between A and Xj (j = i + 1). If two costs are the same, Xi's ID is less than Xj's ID. In another example, X1, X2, ..., Xn are ordered by their IDs. If they are not ordered, then make them in the order.

4. Add Xi (i = 1, 2, ..., n) into the front of Cq and goto step 1.

A third algorithm for building a tree from node R as root starts with a candidate list Cq containing R associated with cost 0 and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft

2. If all the nodes are on Ft, then return with Ft

3. Suppose that node A is associated with a cost Ca which is the cost from root R to node A, node Xi (i = 1, 2, ..., n) is connected to node A and not in Ft and the cost of the link between A and Xi is LCi (i=1, 2, ..., n). Compute Ci = Ca + LCi, check if Xi is in Cq and if Cxi (cost from R to Xi) < Ci. If Xi is not in Cq, then add Xi with cost Ci into Cq; If Xi is in Cq, then If Cxi > Ci then replace Xi with cost Cxi by Xi with Ci in Cq; If Cxi == Ci then add Xi with cost Ci into Cq.

4. Make sure Cq is in a special order. Suppose that Ai (i=1, 2, ..., m) are the nodes in Cq, Cai is the cost associated with Ai,

and IDi is the ID of Ai. One order is that for any k = 1, 2, ..., m-1, Cak < Caj (j = k+1) or Cak = Caj and IDk < IDj. Goto step 1.

A.2. Algorithms to Build Tree Considering Others

An algorithm for building a tree from node R as root with consideration of others's support for flooding reduction starts with a candidate queue Cq containing R associated with previous hop PH=0 and an empty flooding topology Ft:

1. Remove the first node A that supports flooding reduction from the candidate queue Cq if there is such a node A; otherwise (i.e., if there is not such node A in Cq), then remove the first node A from Cq. Add A into the flooding topology Ft.

2. If Cq is empty or all nodes are on Ft, then return with Ft

3. Suppose that node Xi (i = 1, 2, ..., n) is connected to node A and not in the flooding topology Ft and X1, X2, ..., Xn are in a special order considering whether some of them that support flooding reduction (. For example, X1, X2, ..., Xn are ordered by the cost of the link between A and Xi. The cost of the link between A and Xi is less than that of the link between A and Xj (j = i + 1). If two costs are the same, Xi's ID is less than Xj's ID. The cost of a link is redefined such that 1) the cost of a link between A and Xi both support flooding reduction is much less than the cost of any link between A and Xk where Xk with F=0; 2) the real metric of a link between A and Xi and the real metric of a link between A and Xk are used as their costs for determining the order of Xi and Xk if they all (i.e., A, Xi and Xk) support flooding reduction or none of Xi and Xk support flooding reduction.

4. Add Xi (i = 1, 2, ..., n) associated with previous hop PH=A into the end of the candidate queue Cq, and goto step 1.

Another algorithm for building a tree from node R as root with consideration of others' support for flooding reduction starts with a candidate queue Cq containing R associated with previous hop PH=0 and an empty flooding topology Ft:

1. Remove the first node A that supports flooding reduction from the candidate queue Cq if there is such a node A; otherwise (i.e., if there is not such node A in Cq), then remove the first node A from Cq. Add A into the flooding topology Ft.

2. If Cq is empty or all nodes are on Ft, then return with Ft.

3. Suppose that node Xi (i = 1, 2, ..., n) is connected to node A and not in the flooding topology Ft and X1, X2, ..., Xn are in a special order considering whether some of them support flooding reduction. For example, X1, X2, ..., Xn are ordered by the cost of the link between A and Xi. The cost of the link between A and Xi is less than the cost of the link between A and Xj (j = i + 1). If two costs are the same, Xi's ID is less than Xj's ID. The cost of a link is redefined such that 1) the cost of a link between A and Xi both support flooding reduction is much less than the cost of any link between A and Xk where Xk does not support flooding reduction; 2) the real metric of a link between A and Xi and the real metric of a link between A and Xk are used as their costs for determining the order of Xi and Xk if they all (i.e., A, Xi and Xk) support flooding reduction or none of Xi and Xk supports flooding reduction.

4. Add Xi (i = 1, 2, ..., n) associated with previous hop PH=A into the front of the candidate queue Cq, and goto step 1.

A third algorithm for building a tree from node R as root with consideration of others' support for flooding reduction (using flag F = 1 for support, and F = 0 for not support in the following) starts with a candidate list Cq containing R associated with low order cost Lc=0, high order cost Hc=0 and previous hop ID PH=0, and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft.

2. If all the nodes are on Ft, then return with Ft

3. Suppose that node A is associated with a cost Ca which is the cost from root R to node A, node Xi (i = 1, 2, ..., n) is connected to node A and not in Ft and the cost of the link between A and Xi is LCi (i=1, 2, ..., n). Compute Ci = Ca + LCi, check if Xi is in Cq and if Cxi (cost from R to Xi) < Ci. If Xi is not in Cq, then add Xi with cost Ci into Cq; If Xi is in Cq, then If Cxi > Ci then replace Xi with cost Cxi by Xi with Ci in Cq; If Cxi == Ci then add Xi with cost Ci into Cq.

4. Suppose that node A is associated with a low order cost LCa which is the low order cost from root R to node A and a high order cost HCa which is the high order cost from R to A, node Xi (i = 1, 2, ..., n) is connected to node A and not in the flooding topology Ft and the real cost of the link between A and Xi is Ci (i=1, 2, ..., n). Compute LCxi and HCxi: LCxi = LCa + Ci if both A and Xi have flag F set to one, otherwise LCxi = LCa HCxi = HCa + Ci if A or Xi does not have flag F set to one, otherwise HCxi = HCa If Xi is not in Cq, then add Xi associated with LCxi, HCxi and PH = A

into Cq; If Xi associated with LCxi' and HCxi' and PHxi' is in Cq, then If HCxi' > HCxi then replace Xi with HCxi', LCxi' and PHxi' by Xi with HCxi, LCxi and PH=A in Cq; otherwise (i.e., HCxi' == HCxi) if LCxi' > LCxi , then replace Xi with HCxi', LCxi' and PHxi' by Xi with HCxi, LCxi and PH=A in Cq; otherwise (i.e., HCxi' == HCxi and LCxi' == LCxi) if PHxi' > PH, then replace Xi with HCxi', LCxi' and PHxi' by Xi with HCxi, LCxi and PH=A in Cq.

5. Make sure Cq is in a special order. Suppose that Ai (i=1, 2, ..., m) are the nodes in Cq, HCai and LCai are low order cost and high order cost associated with Ai, and IDi is the ID of Ai. One order is that for any k = 1, 2, ..., m-1, HCak < HCaj (j = k+1) or HCak = HCaj and LCak < LCaj or HCak = HCaj and LCak = LCaj and IDk < IDj. Goto step 1.

A.3. Connecting Leaves

Suppose that we have a flooding topology Ft built by one of the algorithms described above. Ft is like a tree. We may connect k (k >=0) leaves to the tree to have a enhanced flooding topology with more connectivity.

Suppose that there are m (0 < m) leaves directly connected to a node X on the flooding topology Ft. Select k (k <= m) leaves through using a deterministic algorithm or rule. One algorithm or rule is to select k leaves that have smaller or larger IDs (i.e., the IDs of these k leaves are smaller/bigger than the IDs of the other leaves directly connected to node X). Since every node has a unique ID, selecting k leaves with smaller or larger IDs is deterministic.

If k = 1, the leaf selected has the smallest/largest node ID among the IDs of all the leaves directly connected to node X.

For a selected leaf L directly connected to a node N in the flooding topology Ft, select a connection/adjacency to another node from node L in Ft through using a deterministic algorithm or rule.

Suppose that leaf node L is directly connected to nodes Ni (i = 1,2,...,s) in the flooding topology Ft via adjacencies and node Ni is not node N, IDi is the ID of node Ni, and Hi (i = 1,2,...,s) is the number of hops from node L to node Ni in the flooding topology Ft.

One Algorithm or rule is to select the connection to node Nj (1 <= j <= s) such that Hj is the largest among H1, H2, ..., Hs. If there is another node Na (1 <= a <= s) and Hj = Ha, then select the one with smaller (or larger) node ID. That is that if Hj == Ha and IDj < IDa then select the connection to Nj for selecting the one with smaller

node ID (or if Hj == Ha and IDj < IDa then select the connection to Na for selecting the one with larger node ID).

Suppose that the number of connections in total between leaves selected and the nodes in the flooding topology Ft to be added is NLc. We may have a limit to NLc.

Authors' Addresses

Huaimo Chen
Huawei Technologies

Email: huaimo.chen@huawei.com

Dean Cheng
Huawei Technologies

Email: dean.cheng@huawei.com

Mehmet Toy
Verizon
USA

Email: mehmet.toy@verizon.com

Yi Yang
IBM
Cary, NC
United States of America

Email: yyietf@gmail.com

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

U. Chunduri
R. Li
Huawei USA
R. White
LinkedIn
J. Tantsura
Nuage Networks
L. Contreras
Telefonica
Y. Qu
Huawei USA
July 2, 2018

Preferred Path Routing (PPR) in IS-IS
draft-chunduri-lsr-isis-preferred-path-routing-01

Abstract

This document specifies a Preferred Path Routing (PPR) mechanism to simplify the path description of data plane traffic in Segment Routing (SR) deployments. PPR aims to mitigate the MTU and data plane processing issues that may result from SR packet overheads; and also supports traffic measurement, accounting statistics and further attribute extensions along the paths. Preferred Path Routing is achieved through the addition of descriptions to IS-IS advertised prefixes, and mapping those to a PPR data-plane identifier.

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 [RFC2119], RFC8174 [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 January 3, 2019.

Table of Contents

1. Introduction

In a network implementing Segment Routing (SR), packets are steered through the network using Segment Identifiers (SIDs) carried in the packet header. Each SID uniquely identifies a segment as defined in [I-D.ietf-spring-segment-routing]. SR capabilities are defined for MPLS and IPv6 data planes called SR-MPLS and SRv6 respectively.

In SR-MPLS, each segment is encoded as a label, and an ordered list of segments are encoded as a stack of labels. This stack of labels is carried as part of the packet header. In SRv6, a segment is encoded as an IPv6 address, within a new type of IPv6 hop-by-hop routing header/extension header (EH) called SRH [I-D.ietf-6man-segment-routing-header]; an ordered list of IPv6 addresses/segments are encoded in SRH.

Section 1.2 and Section 1.3 describe performance, hardware capabilities and various associated issues which may result in SR deployments. These motivate the proposed solution, Preferred Path Routing, which is specified in Section 2.

1.1. Acronyms

EL	-	Entropy Label
ELI	-	Entropy Label Indicator
LSP	-	IS-IS Link State PDU
MPLS	-	Multi Protocol Label Switching
MSD	-	Maximum SID Depth
MTU	-	Maximum Transferrable Unit
PPR	-	Preferred Path Routing/Route
PPR-ID	-	Preferred Path Route Identifier, a data plane identifier
SID	-	Segment Identifier
SPF	-	Shortest Path First
SR-MPLS	-	Segment Routing with MPLS data plane
SRH	-	Segment Routing Header - IPv6 routing Extension headr
SRv6	-	Segment Routing with Ipv6 data plane with SRH

TE - Traffic Engineering

1.2. Challenges with Increased SID Depth

SR label stacks carried in the packet header create challenges in the design and deployment of networks and networking equipment. Following examples illustrates the need for increased SID depth in various use cases:

(a). Consider the following network where SR-MPLS data plane is in use and with same SRGB (5000-6000) on all nodes i.e., A1 to A7 and B1 to B7 for illustration:

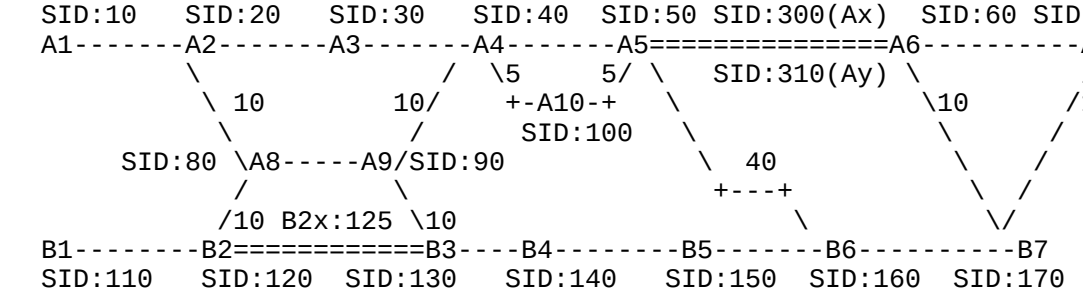


Figure 1: SR-MPLS Network

Global ADJ SIDs are provisioned between A5-A6 and B2-B3. All other SIDs shown are nodal SID indices.

All metrics of the links are set to 1, unless marked otherwise.

Shortest Path from A1 to A7: A2-A3-A4-A5-A6-A7

Path-x: From A1 to A7 - A2-A8-B2-B2x-A9-A10-Ax-A7; Pushed Label Stack @A1: 5020:5080:5120:5125:5090:5100:5300:5070 (where B2x is a local ADJ-SID and Ax is a global ADJ-SID).

In this example, the traffic engineered path is represented with a combination of Adjacency and Node SIDs with a stack of 8 labels. However, this value can be larger, if the use of entropy label [RFC6790] is desired and based on the Readable Label Depth (Section 1.3) capabilities of each node and additional labels required to insert ELI/EL at appropriate places.

Though above network is shown with SR-MPLS data plane, if the network were to use SR-IPv6 data plane, path size would be

increased even more because of the size of the IPv6 SID (16 bytes) in SRH.

(b). Apart from the TE case above, when deploying [I-D.ietf-mpls-sfc] or [I-D.xuclad-spring-sr-service-chaining], with the inclusion of services, or non-topological segments on the label stack, can also make the size of the stack much larger.

(c). Some SR-MPLS deployments need accounting statistics for path monitoring and traffic re-optimizations. [I-D.hegde-spring-traffic-accounting-for-sr-paths] and [I-D.cheng-spring-mpls-path-segment] propose solutions with various forms of path segments (either with special labels or PATH segment encoded at the bottom of the stack respectively). However, these proposals further increases the depth of SID stack, when it is compounded with MSD/RLDs of various nodes in the path.

Overall the additional path overhead in various SR deployments may cause the following issues:

a. HW Capabilities: Not all nodes in the path can support the ability to push or read label stack needed [I-D.ietf-isis-segment-routing-msd] to satisfy user/operator requirements, Alternate paths which meet these user/operator requirements may not be available.

b. Line Rate: Potential performance issues in deployments, which use SRH data plane with the increased size of the SRH with 16 byte SIDs.

c. MTU: Larger SID stacks on the data packet can cause potential MTU/fragmentation issues.

d. Header Tax: Some deployments, such as 5G, require minimal packet overhead in order to conserve network resources. Carrying 40 or 50 octets of data in a packet with hundreds of octet of header would be an unacceptable use of available bandwidth.

With the solution proposed in this document (Section 5) and Section 4), for Path-x in the example network Figure 1 above, SID stack would be reduced from 8 SIDs to a single SID.

1.3. Mitigation with MSD

The number of SIDs in the stack a node can impose is referred as Maximum SID Depth (MSD) capability [I-D.ietf-isis-segment-routing-msd], which must be taken into consideration when computing a path to transport a data packet in a

network implementing segment routing. [I-D.ietf-isis-mpls-elc] defines another MSD type, Readable Label Depth (RLD) that is used by a head-end to insert Entropy Label pair (ELI/EL) at appropriate depth, so it could be read by transit nodes. There are situations where the source routed path can be excessive as path represented by SR SIDs need to describe all the nodes and ELI/EL based on the readability of the nodes in that path.

MSDs (and RLD type) capabilities advertisement help mitigate the problem for a central entity to create the right source routed path per application/operator requirements. However the availability of actual paths meeting these requirements are still limited by the underlying hardware and their MSD capabilities in the data path.

2. Preferred Path Routing (PPR)

PPR mitigates the issues described in Section 1.2, while continuing to allow the direction of traffic along an engineered path through the network by replacing the label stack with a PPR-ID. The PPR-ID can either be a single label or a native destination address. To facilitate the use of a single label to describe an entire path, a new TLV is added to IS-IS, as described below in Section 3.

A PPR could be an SR path, a traffic engineered path computed based on some constraints, an explicitly provisioned Fast Re-Route (FRR) path or a service chained path. A PPR can be signaled by any node, computed by a central controller, or manually configured by an operator. PPR extends the source routing and path steering capabilities to native IP (IPv4 and IPv6) data planes without hardware upgrades; see Section 5.

2.1. PPR-ID and PPR Path Description

The PPR-ID describes a path through the network. For SR- MPLS this is an MPLS Label/SID and for SRv6 this is an IPv6-SID. For native IP data planes this is either IPv4 or IPv6 address/prefix.

The path identified by the PPR-ID is described as a set of Path Description Elements (PDEs), each of which represents a segment of the path. Each node determines location in the path as described, and forwards to the next segment/hop or label of the path description (see the Forwarding Procedure Example later in this document).

These PPR-PDEs as defined in Section 3.3, like SR SIDs, can represent topological elements like links/nodes, backup nodes, as well as non-topological elements such as a service, function, or context on a particular node. PPR-PDE optionally, can also have more information as described with in their Sub-TLVs.

A PPR path can be described as a Strict-PPR or a Loose-PPR. In a Strict-PPR all nodes/links on the path are described with SR SIDs for SR data planes or IPv4/IPV6 addresses for native IP data planes. In a Loose-PPR only some of the nodes/links from source to destination are described. More specifics and restrictions around Strict/Loose PPRs are described in respective data planes in Section 5. Each PDE is described as either an MPLS label towards the next hop in MPLS enabled networks, or as an IP next hop, in the case of either "plain"/"native" IP or SRv6 enabled networks. A PPR path is related to a set of PDEs using the following TLVs.

3. PPR Related TLVs

This section describes the encoding of PPR TLV. This TLV can be seen as having 4 logical section viz., encoding of the PPR-Prefix (IS-IS Prefix), encoding of PPR-ID, encoding of path description with an ordered PDE Sub-TLVs and a set of optional PPR attribute Sub-TLVs, which can be used to describe one or more parameters of the path. Multiple instances of this TLV MAY be advertised in IS-IS LSPs with different PPR-ID Type and with corresponding PDE Sub-TLVS. The PPR TLV has Type TBD (suggested value xxx), and has the following format:

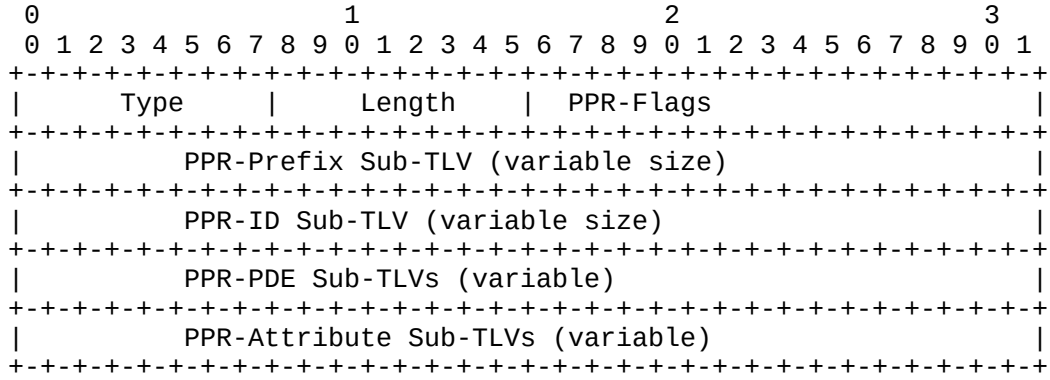


Figure 2: PPR TLV Format

- Type: TBD (IANA) from IS-IS top level TLV registry.
- Length: Total length of the value field in bytes.
- PPR-Flags: 2 Octet bit-field of flags for this TLV; described below.
- PPR-Prefix: A variable size sub-TLV representing the destination of the path being described. This is defined in Section 3.1.

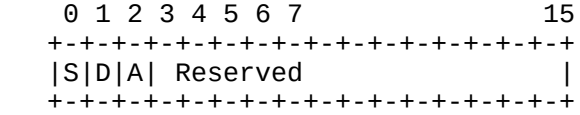
- o PPR-ID: A variable size Sub-TLV representing the data plane or forwarding identifier of the PPR. Defined in Section 3.2.

- o PPR-PDEs: Variable number of ordered PDE Sub-TLVs which represents the path. This is defined in Section 3.3.

- o PPR-Attributes: Variable number of PPR-Attribute Sub-TLVs which represent the path attributes. These are defined in Section 3.4.

The Flags field has the following flag bits defined:

PPR TLV Flags Format



1. S: If set, the PPR TLV MUST be flooded across the entire routing domain. If the S flag is not set, the PPR TLV MUST NOT be leaked between IS-IS levels. This bit MUST NOT be altered during the TLV leaking

2. D: When the PPR TLV is leaked from IS-IS level-2 to level-1, the D bit MUST be set. Otherwise, this bit MUST be clear. PPR TLVs with the D bit set MUST NOT be leaked from level-1 to level-2. This is to prevent TLV looping across levels.

3. A: The originator of the PPR TLV MUST set the A bit in order to signal that the prefixes and PPR-IDs advertised in the PPR TLV are directly connected to the originators. If this bit is not set, this allows any other node in the network advertise this TLV on behalf of the originating node of the PPR-Prefix. If PPR TLV is leaked to other areas/levels the A-flag MUST be cleared. In case if the originating node of the prefix must be disambiguated for any reason including, if it is a Multi Homed Prefix (MHP) or leaked to a different IS-IS level or because [RFC7794] X-Flag is set, then PPR-Attribute Sub-TLV Source Router ID SHOULD be included.

4. Reserved: For future use; MUST be set to 0 on transmit and ignored on receive.

The following sub-TLVs draw from a new registry for sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process.

3.1. PPR-Prefix Sub-TLV

The structure of PPR-Prefix is:

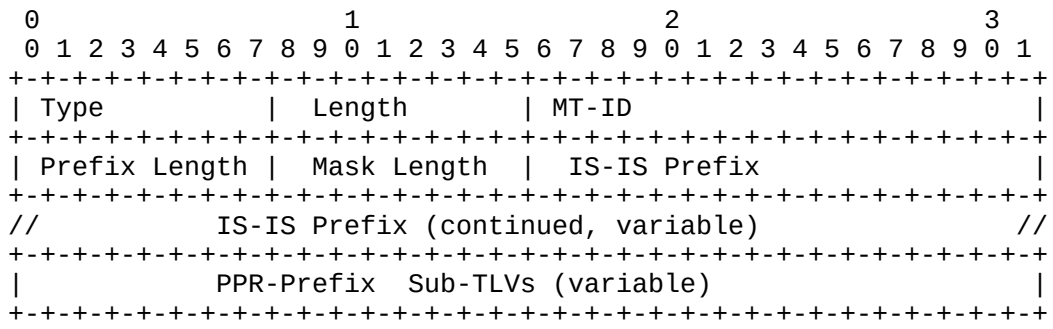


Figure 3: PPR-Prefix Sub-TLV Format

- Type: TBD (IANA to assign from sub-TLV registry described above).
- Length: Total length of the value field in bytes.
- MT-ID: The multi-topology identifier defined in [RFC5120]; the 4 most significant bits MUST be set to 0 on transmit and ignored on receive. The remaining 12-bit field contains the MT-ID.
- Prefix Length: The length of the prefix in bytes. For IPv4 it MUST be 4 and IPv6 it MUST be 16 bytes.
- Mask Length: The length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- IS-IS Prefix: The IS-IS prefix at the tail-end of the advertised PPR. This corresponds to a routable prefix of the originating node and it MAY have one of the [RFC7794] flags set (X-Flag/R-Flag/N-Flag). Value of this field MUST be 4 octets for IPv4 Prefix and MUST be 16 octets for IPv6 Prefix. Encoding is similar to TLV 135 [RFC5305] and TLV 236 [RFC5308] or MT-Capable [RFC5120] IPv4 (TLV 235) and IPv6 Prefixes (TLV 237) respectively.
- PPR-Prefix Sub-TLVs - TBD. These have 1 octet type, 1 octet length and value field is defined per the type field.

3.2. PPR-ID Sub-TLV

This is the actual data plane identifier in the packet header and could be of any data plane as defined in PPR-ID Type field. Both PPR-Prefix and PPR-ID MUST belong to a same node in the network.

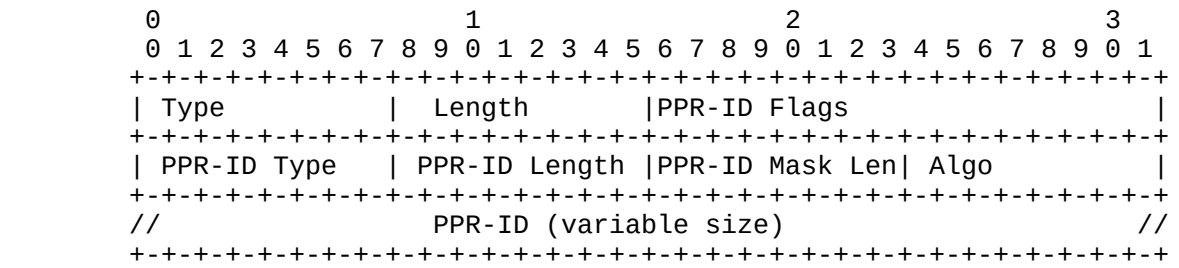
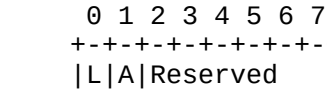


Figure 4: PPR-ID Sub-TLV Format

- Type: TBD (IANA to assign from sub-TLV registry described above).
- Length: Total length of the value field in bytes.
-
  - PPR-ID Flags: 2 Octet field for PPR-ID flags:
-

PPR-ID Flags Format



-

1. L: If set, the PPR path is a Loose-PPR. If the this flag is unset, then the PPR path is a Strict-PPR. A Strict-PPR lists every single node or adjacency in the path description from source to the destination.
2. A: If set, all non-PPR path nodes in the IS-IS area/domain MUST add a FIB entry for the PPR-ID with NH set to the shortest path NH for the prefix being advertised. The use of this is TBD. By default this flag MUST be unset.
3. Reserved: For future use; MUST be set to 0 on transmit and ignored on receive.

-

  * PPR-ID Type: Data plane type of PPR-ID. This is a new registry (TBD IANA - Suggested values as below) for this Sub-TLV and the defined types are as follows:

o

A. Type: 1 MPLS SID/Label

B. Type: 2 Native IPv4 Address/Prefix

C. Type: 3 Native IPv6 Address/Prefix

D. Type: 4 IPv6 SID in SRv6 with SRH

- PPR-ID Length: Length of the PPR-ID field in octets and this depends on the PPR-ID type. See PPR-ID below for the length of this field and other considerations.

- PPR-ID Mask Length: It is applicable for only for PPR-ID Type 2, 3 and 4. For Type 1 this value MUST be set to zero. It contains the length of the PPR-ID Prefix in bits. Only the most significant octets of the Prefix are encoded. This is needed, if PPR-ID followed is an IPv4/IPv6 Prefix instead of 4/16 octet Address respectively.

- Algo: 1 octet value represents the SPF algorithm. Algorithm registry is as defined in [I-D.ietf-isis-segment-routing-extensions].

- PPR-ID: This is the Preferred Path forwarding identifier that would be on the data packet. The value of this field is variable and it depends on the PPR-ID Type - for Type 1, this is and MPLS SID/Label. For Type 2 this is a 4 byte IPv4 address. For Type 3 this is a 16 byte IPv6 address. For Type 2 and Type 3 encoding is similar to "IS-IS Prefix" as specified in Section 3.1. For Type 4, it is a 16 byte IPv6 SID.

For PPR-ID Type 2, 3 or 4, if the PPR-ID Len is set to non-zero value, then the PPR-ID MUST NOT be advertised as a routable prefix in TLV 135, TLV 235, TLV 236 and TLV 237. Also PPR-ID MUST belong to the node where Prefix is advertised from. PPR-ID Len = 0 case is a special case and is discussed in Section 4.1.

3.3. PPR-PDE Sub-TLV

This Sub-TLV represents the PPR Path Description Element (PDE). PPR-PDEs are used to describe the path in the form of set of contiguous and ordered Sub-TLVs, where first Sub-TLV represents (the top of the

stack in MPLS data plane or) first node/segment of the path. These set of ordered Sub-TLVs can have both topological elements and non-topological elements (e.g., service segments).

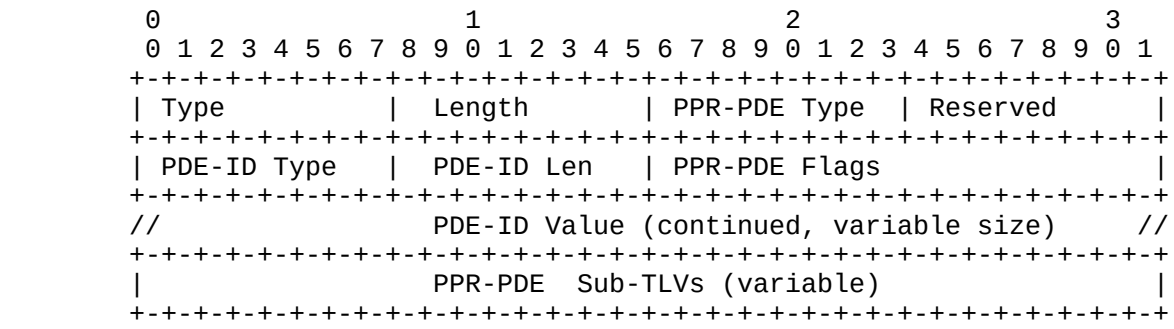


Figure 5: PPR-PDE Sub-TLV Format

- Type: TBD (See IANA for suggested value) from IS-IS PPR TLV Section 3 Sub-TLV registry.
- Length: Total length of the value field in bytes.
- PPR-PDE Type: A new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 Topological

b. Type: 2 Non-Topological

- PDE-ID Type: 1 Octet PDE-forwarding IDentifier Type. A new registry (TBD IANA) for this Sub-TLV and the defined types and corresponding PDE-ID Len, PDE-ID Value are as follows:

a. Type 1: SID/Label type as defined in [I-D.ietf-isis-segment-routing-extensions]. PDE-ID Len and PDE-ID Value fields are per Section 2.3 of the referenced document.

b. Type 2: SR-MPLS Prefix SID. PDE-ID Len and PDE-ID Value are same as Type 1.

c. Type 3: SR-MPLS Adjacency SID. PDE-ID Len and PDE-ID Value are same as Type 1.

d. Type 4: IPv4 Address. PDE-ID Len is 4 bytes and PDE-ID Value is 4 bytes IPv4 address encoded similar to IPv4 Prefix described in Section 3.1.

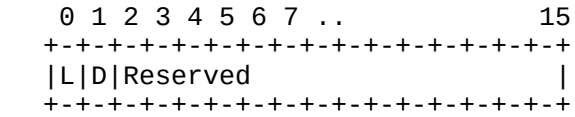
e. Type 5: IPv6 Address. PDE-ID Len is 16 bytes and PDE-ID Value is 16 bytes IPv6 address encoded similar to IPv6 Prefix described in Section 3.1.

f. Type 6: SRv6 Node SID as defined in [I-D.bashandy-isis-srv6-extensions]. PDE-ID Len and PDE-ID Value are as defined in SRv6 SID from the refrenced draft.

g. Type 7: SRv6 Adjacency-SID. PDE-ID Len and PDE-ID Values are similar to SRv6 Node SID above.

o PPR-PDE Flags: 2 Octet bit-field of flags; described below:

PPR-PDE Flags Format



1. L: Loose Bit. Indicates the type of next "Topological PDE-ID" in the path description and overrides the L bit in Section 3.2. If set, the next PDE is Loose. If this flag is unset, the next Topological PDE is Strict Type.

2. D: Destination bit. By default this bit MUST be unset. This bit MUST be set only for PPR-PDE Type is 1 i.e., Topological and this PDE represents the node PPR-Prefix Section 3.1 belongs to, if there is no sub-sub-TLV to override PPR-Prefix and PPR-ID values.

3. Reserved: 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

o PPR-PDE Sub-TLVs: TBD. These have 1 octet type, 1 octet length and value field is defined per the type field.

3.4. PPR-Attributes Sub-TLV

PPR-Attribute Sub-TLVs describe the attributes of the path. The following sub-TLVs draw from a new registry for sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process:

o Type 1 (Suggested Value - IANA TBD): Packet Traffic accounting Sub-TLV. Length 0 and no value field. Specifies to create a counter to count number of packets forwarded on this PPR-ID on each node in the path description.

- Type 2 (Suggested Value - IANA TBD): Traffic statistics in Bytes Sub-TLV. Length 0 and no value field. Specifies to create a counter to count number of bytes forwarded on this PPR-ID specified in the network header (e.g. IPv4, IPv6) on each node in the path description.

- Type 3 (Suggested Value - IANA TBD): PPR-Prefix originating node's IPv4 Router ID Sub-TLV. Length and Value field are as specified in [RFC7794].

- Type 4 (Suggested Value - IANA TBD): PPR-Prefix originating node's IPv6 Router ID Sub-TLV. Length and Value field are as specified in [RFC7794].

- Type 5 (Suggested Value - IANA TBD): PPR-Metric Sub-TLV. Length 4 bytes, and Value is metric of this path represented through the PPR-ID. Different nodes can advertise the same PPR-ID for the same Prefix with a different set of PPR-PDE Sub-TLVs and the receiving node MUST consider the lowest metric value (TBD more, on what happens when metric is same for two different set of PPR-PDE Sub-TLVs).

4. PPR Processing Procedure Example

As specified in Section 2, a PPR can be a TE path, locally provisioned by the operator or by a controller. Consider the following IS-IS network to describe the operation of PPR TLV as defined in Section 3:

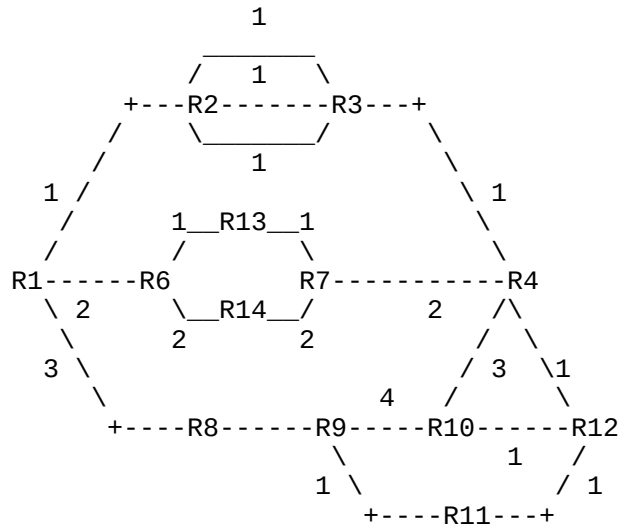


Figure 6: IS-IS Network

In the (Figure 6) shown, consider node R1 as an ingress node, or a head-end node, and the node R4 MAY be an egress node or another head-end node. The numbers shown on links between nodes R1-R14 indicate the bi-directional IS-IS metric as provisioned. R1 may be configured to receive TE source routed path information from a central entity (PCE [RFC5440], Netconf [RFC6241] or a Controller) that comprises of PPR information which relates to sources that are attached to R1. It is also possible to have a PPR provisioned locally by the operator for non-TE needs (FRR or for chaining certain services).

The PPR TLV (as specified in Section 3) is encoded as an ordered list of PPR-PDEs from source to a destination node in the network and is represented with a PPR-ID Section 3.2. The PPR TLV includes PPR-PDE Sub-TLVS Section 3.3, which represent both topological and non-topological elements and specifies the actual path towards a PPR-Prefix at R4.

- The shortest path towards R4 from R1 are through the following sequence of nodes: R1-R2-R3-R4 based on the provisioned metrics.

- The central entity can define a few PPRs from R1 to R4 that deviate from the shortest path based on other network characteristic requirements as requested by an application or service. For example, the network characteristics or performance requirements may include bandwidth, jitter, latency, throughput, error rate, etc.

- o A first PPR may be identified by PPR-ID = 1 (value) and may include the path of R1-R6-R7-R4 for a Prefix advertised by R4. This is an example for a Loose-PPR and 'L' bit MUST be set on Section 3.2.

- o A second PPR may be identified by PPR-ID = 2 (value) and may include the path of R1-R8-R9-R10-R4. This is an example for a Strict-PPR and 'L' bit MUST be unset on Section 3.2. Though this example shows PPR with all nodal SIDs, it is possible to have a PPR with combination of node and adjacency SIDs (local or global) or with PPR-PDE Type set to Non-Topological as defined in Section 3.3 elements along with these.

4.1. PPR TLV Processing

The first topological sub-object or PDE (Section 3.3) relative to the beginning of PPR Path contains the information about the first node (e.g. in SR-MPLS it's the topmost label). The last topological sub-object or PDE contains information about the last node (e.g. in SR-MPLS it's the bottommost label).

Each receiving node, determine whether an advertised PPR includes information regarding the receiving node. Before processing any further, validation MUST be done to see if any PPR topological PDE is seen more than once (possible loop), if yes, this PPR TLV MUST be ignored. Processing of PPR TLVs can be done, during the end of the SPF computation (for MTID that is advertised in this TLV) and for the each prefix described through PPR TLV. For example, node R9 receives the PPR information, and ignores the PPR-ID=1 (Section 4) because this PPR TLV does not include node R9 in the path description/ordered PPR-PDE list.

However, node R9 may determine that the second PPR identified by PPR-ID = 2 does include the node R9 in its PDE list. Therefore, node R9 updates the local forwarding database to include an entry for the destination address of R4 indicates, that when a data packet comprising a PPR-ID of 2 is received, forward the data packet to node R10 instead of R11. This is even though from R9 the shortest path to reach R4 via R11 (Cost 3: R9-R11-R12-R4) it chooses the nexthop to R10 to reach R4 as specified in the PPR path description. Same process happens to all nodes or all topological PDEs as described in the PPR TLV.

In summary, the receiving node checks first, if this node is on the path by checking the node's topological elements (with PPR-PDE Type set to Topological) in the path list. If yes, it adds/adjusts the shortest path nexthop computed towards PPR Prefix to the shortest path nexthop towards the next topological PDE in the PPR's Path.

For PPR-ID (Section 3.2) Type 2, 3 or 4, if the PPR-ID Len is set to 0, then Prefix would also become the PPR-ID (a special case). This can be used for some situations, where certain optimizations are required in the network. For this, path described in the PPR TLV SHOULD be completely dis-joint from the shortest path route to the prefix. If the disjoint-ness property is not maintained then the traffic MAY not be using the PPR path, as and when it encounters any node which is not in the path description.

5. PPR Data Plane aspects

Data plane for PPR-ID is selected by the entity (e.g., a controller, locally provisioned by operator), which selects a particular PPR in the network. Section 3.2 defines various data plane identifier types and a corresponding data plane identifier is selected by the entity which selects the PPR. Other data planes other than described below can also use this TLV to describe the PPR. Further details TBD.

5.1. SR-MPLS with PPR

If PPR-ID Type is 1, then the PPR belongs to SR-MPLS data plane and the complete PPR stack is represented with a unique SR SID/Label and this gets programmed on the data plane of each node, with the appropriate nexthop computed as specified in Section 4. PPR-ID here is a label/index from the SRGB (like another node SID or global-ADJ SID). PPR path description here is a set of ordered SIDs represented with PPR-PDE (Section 3.2) Sub-TLVs. Non-Topological segments also programmed in the forwarding to enable specific function/service, when the data packet hits with corresponding PPR-ID.

Based on 'L' flag in PPR-ID Flags (Section 3.2), for SR-MPLS data plane either 1 label or 2 labels need to be provisioned on individual nodes on the path description. For the example network in Section 4, for PPR-ID=1, which is a loose path, node R6 programs the bottom label as PPR-ID and the top label as the next topological PPR-PDE in the path, which is a node SID of R7. The NH computed at R6 would be the shortest path towards R7 i.e., the interface towards R13. If 'L' flag is unset only PPR-ID is programmed on the data plane with NH set to the shortest path towards next topological PPR-PDE.

5.2. SRv6 with PPR

If PPR-ID Type is 4, the PPR belongs to SRv6 with SRH data plane and the complete PPR stack is represented with IPv6 SIDs and this gets programmed on the data plane with the appropriate nexthop computed as specified in Section 4. PPR-ID here is a SRv6 SID. PPR path description here is a set of ordered SID TLVs similar to as specified in Section 5.1. One way PPR-ID would be used in this case is by

setting the same as the destination IPv6 address and SL field in SRH would be set to 0; however SRH [I-D.ietf-6man-segment-routing-header] can contain any other TLVs and non-topological SIDs as needed.

5.3. PPR Native IP Data Planes

If PPR-ID Type is 2 then source routing and packet steering can be done in IPv4 data plane (PPR-IPv4), along the path as described in PPR Path description. This is achieved by setting the destination IP address as PPR-ID, which is an IPv4 address in the data packet (tunneled/encapsulated). There is no data plane change or upgrade needed to support this. However this is applicable to only Strict-PPR paths ('L' bit as specified in Section 3.2 MUST be unset).

Similarly for PPR-ID Type is 3, then source routing and packet steering can be done in IPv6 data plane (PPR-IPv6), along the path as described in PPR Path description. Whatever specified above for IPv4 applies here too, except that destination IP address of the data packet is IPv6 Address (PPR-ID). This doesn't require any IPv6 extension headers (EH), if there is no metadata/TLVs need to be carried in the data packet.

6. PPR Scaling Considerations

In a network of N nodes O(N^2) total (unidirectional) paths are necessary to establish any-to-any connectivity, and multiple (k) such path sets may be desirable if multiple path policies are to be supported (lowest latency, highest throughput etc.).

In many solutions and topologies, N may be small enough and/or only a small set of paths need to be preferred paths, for example for high value traffic (DetNet, some of the defined 5G slices), and then the technology specified in this document can support these deployments.

However, to address the scale needed when a larger number of PPR paths are required, the PPR TREE structure can be used [I-D.draft-ce-ppr-graph-00]. Each PPR Tree uses one label/SID and defines paths from any set of nodes to one destination, thus reduces the number of entries needed in SRGB at each node (for SR-MPLS data plane Section 5.1). These paths form a tree rooted in the destination. In other word, PPR Tree identifiers are destination identifiers and PPR Treed are path engineered destination routes (like IP routes) and it scaling simplifies to linear in N i.e., O(k*N).

7. PPR Traffic Accounting

Section 3.4 defines few PPR-Attributes to indicate creation of traffic accounting statistics in each node of the PPR path description. Presence of "Packet Traffic Accounting" and "Traffic Statistics" Sub-TLVs instruct to provision the hardware, to account for the respective traffic statistics. Traffic accounting should happen, when the actual data traffic hits for the PPR-ID in the forwarding plane. This allows more granular and dynamic enablement of traffic statistics for only certain PPRs as needed.

This approach, thus is more safe and secure than any mechanism that involves creation of the state in the nodes with the data traffic itself. This is because, creation and deletion of the traffic accounting state for PPRs happen through IS-IS LSP processing and IS-IS protocol control plane security Section 10 options are applicable to this TLV.

How the traffic accounting is distributed to a central entity is out of scope of this document. One can use any method (e.g. gRPC) to extract the PPR-ID traffic stats from various nodes along the path.

8. Acknowledgements

Thanks to Alex Clemm, Lin Han, Toerless Eckert and Kiran Makhijani for initial discussions on this topic. Thanks to Kevin Smith and Stephen Johnson for various deployment scenarios applicability from ETSI WGs perspective. Authors also acknowledge Alexander Vainshtein for detailed discussions and suggestions on this topic.

Earlier versions of draft-ietf-isis-segment-routing-extensions have a mechanism to advertise EROs through Binding SID.

9. IANA Considerations

This document requests the following new TLV in IANA IS-IS TLV code-point registry.

TLV #	Name
TBD	PPR TLV

This document requests IANA to create a new Sub-TLV registry for PPR TLV Section 3 with the following initial entries (suggested values):

Sub-TLV #	Sub-TLV Name

1	PPR-Prefix (Section 3.1)
2	PPR-ID (Section 3.2)
3	PPR-PDE (Section 3.3)

This document also requests IANA to create a new Sub-TLV registry for PPR Path attributes with the following initial entries (suggested values):

Sub-TLV #	Sub-TLV Name
1	Packet Traffic Accounting (Section 3.4)
2	Traffic Statistics (Section 3.4)
3	PPR-Prefix Source IPv4 Router ID (Section 3.4)
4	PPR-Prefix Source IPv6 Router ID (Section 3.4)
5	PPR-Metric (Section 3.4)

This document requests additional IANA registries in an IANA managed registry "Interior Gateway Protocol (IGP) Parameters" for various PPR TLV parameters. The registration procedure is based on the "Expert Review" as defined in [RFC8126]. The suggested registry names are:

- "PPR-Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3 of this document.
- "PPR-Flags" - 1 Octet. Bits as described in Section 3 of this document.
- "PPR-ID Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.2 of this document.
- "PPR-ID Flags" - 1 Octet. Bits as described in Section 3.2 of this document.
- "PPR-PDE Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.3 of this document.
- "PPR-PDE Flags" - 1 Octet. Bits as described in Section 3.3 of this document.
- "PDE-ID Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.3 of this document.

10. Security Considerations

Security concerns for IS-IS are addressed in [RFC5304] and [RFC5310]. Further security analysis for IS-IS protocol is done in [RFC7645] with detailed analysis of various security threats and why [RFC5304] should not be used in the deployments. Advertisement of the additional information defined in this document introduces no new security concerns in IS-IS protocol. However as this extension is related to SR-MPLS and SRH data planes as defined in [I-D.ietf-spring-segment-routing], those particular data plane security considerations does apply here.

11. References

11.1. Normative References

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-12 (work in progress), May 2018.

[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.

[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>.

11.2. Informative References

[I-D.bashandy-isis-srv6-extensions]
Ginsberg, L., Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-03 (work in progress), June 2018.

[I-D.cheng-spring-mpls-path-segment]
Cheng, W., Wang, L., Li, H., Chen, M., Zigler, R., and S. Zhan, "Path Segment in MPLS Based Sement Routing Network", draft-cheng-spring-mpls-path-segment-01 (work in progress), March 2018.

[I-D.hegde-spring-traffic-accounting-for-sr-paths]
Hegde, S., "Traffic Accounting for MPLS Segment Routing Paths", draft-hegde-spring-traffic-accounting-for-sr-paths-01 (work in progress), October 2017.

[I-D.ietf-6man-segment-routing-header]
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-14 (work in progress), June 2018.

[I-D.ietf-isis-mpls-elc]
Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability and Readable Label-stack Depth Using IS-IS", draft-ietf-isis-mpls-elc-03 (work in progress), January 2018.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-18 (work in progress), June 2018.

[I-D.ietf-mpls-sfc]
Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", draft-ietf-mpls-sfc-01 (work in progress), May 2018.

[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.

[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>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[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>.

[RFC7645] Chunduri, U., Tian, A., and W. Lu, "The Keying and Authentication for Routing Protocol (KARP) IS-IS Security Analysis", RFC 7645, DOI 10.17487/RFC7645, September 2015, <https://www.rfc-editor.org/info/rfc7645>.

[RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <https://www.rfc-editor.org/info/rfc7794>.

[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>.

[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>.

Authors' Addresses

Uma Chunduri
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: uma.chunduri@huawei.com

Richard Li
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: renwei.li@huawei.com

Russ White
LinkedIn
Oak Island, NC 28465
USA

Email: russ@riw.us

Jeff Tantsura
Nuage Networks
755 Ravendale Drive
Mountain View, CA 94043
USA

Email: jefftant.ietf@gmail.com

Luis M. Contreras
Telefonica
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

Yingzhen Qu
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yingzhen.qu@huawei.com

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

U. Chunduri
R. Li
Huawei USA
R. White
LinkedIn
J. Tantsura
Nuage Networks
L. Contreras
Telefonica
Y. Qu
Huawei USA
July 2, 2018

Preferred Path Routing (PPR) in IS-IS
draft-chunduri-lsr-isis-preferred-path-routing-01

Abstract

This document specifies a Preferred Path Routing (PPR) mechanism to simplify the path description of data plane traffic in Segment Routing (SR) deployments. PPR aims to mitigate the MTU and data plane processing issues that may result from SR packet overheads; and also supports traffic measurement, accounting statistics and further attribute extensions along the paths. Preferred Path Routing is achieved through the addition of descriptions to IS-IS advertised prefixes, and mapping those to a PPR data-plane identifier.

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 [RFC2119], RFC8174 [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 January 3, 2019.

Table of Contents

1. Introduction

In a network implementing Segment Routing (SR), packets are steered through the network using Segment Identifiers (SIDs) carried in the packet header. Each SID uniquely identifies a segment as defined in [I-D.ietf-spring-segment-routing]. SR capabilities are defined for MPLS and IPv6 data planes called SR-MPLS and SRv6 respectively.

In SR-MPLS, each segment is encoded as a label, and an ordered list of segments are encoded as a stack of labels. This stack of labels is carried as part of the packet header. In SRv6, a segment is encoded as an IPv6 address, within a new type of IPv6 hop-by-hop routing header/extension header (EH) called SRH [I-D.ietf-6man-segment-routing-header]; an ordered list of IPv6 addresses/segments are encoded in SRH.

Section 1.2 and Section 1.3 describe performance, hardware capabilities and various associated issues which may result in SR deployments. These motivate the proposed solution, Preferred Path Routing, which is specified in Section 2.

1.1. Acronyms

EL	-	Entropy Label
ELI	-	Entropy Label Indicator
LSP	-	IS-IS Link State PDU
MPLS	-	Multi Protocol Label Switching
MSD	-	Maximum SID Depth
MTU	-	Maximum Transferrable Unit
PPR	-	Preferred Path Routing/Route
PPR-ID	-	Preferred Path Route Identifier, a data plane identifier
SID	-	Segment Identifier
SPF	-	Shortest Path First
SR-MPLS	-	Segment Routing with MPLS data plane
SRH	-	Segment Routing Header - IPv6 routing Extension headr
SRv6	-	Segment Routing with Ipv6 data plane with SRH

TE - Traffic Engineering

1.2. Challenges with Increased SID Depth

SR label stacks carried in the packet header create challenges in the design and deployment of networks and networking equipment. Following examples illustrates the need for increased SID depth in various use cases:

(a). Consider the following network where SR-MPLS data plane is in use and with same SRGB (5000-6000) on all nodes i.e., A1 to A7 and B1 to B7 for illustration:

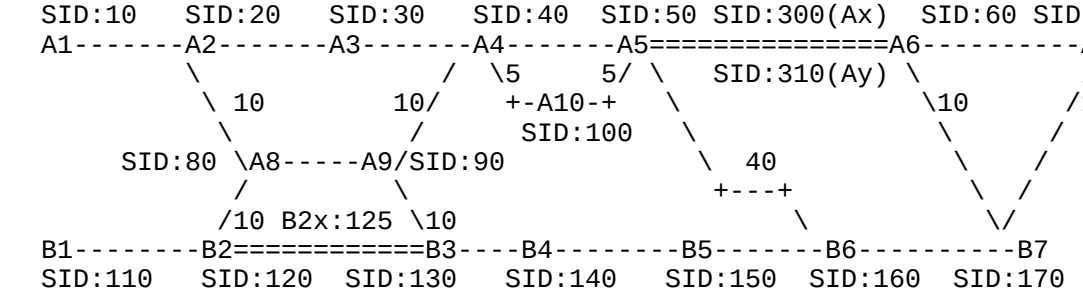


Figure 1: SR-MPLS Network

Global ADJ SIDs are provisioned between A5-A6 and B2-B3. All other SIDs shown are nodal SID indices.

All metrics of the links are set to 1, unless marked otherwise.

Shortest Path from A1 to A7: A2-A3-A4-A5-A6-A7

Path-x: From A1 to A7 - A2-A8-B2-B2x-A9-A10-Ax-A7; Pushed Label Stack @A1: 5020:5080:5120:5125:5090:5100:5300:5070 (where B2x is a local ADJ-SID and Ax is a global ADJ-SID).

In this example, the traffic engineered path is represented with a combination of Adjacency and Node SIDs with a stack of 8 labels. However, this value can be larger, if the use of entropy label [RFC6790] is desired and based on the Readable Label Depth (Section 1.3) capabilities of each node and additional labels required to insert ELI/EL at appropriate places.

Though above network is shown with SR-MPLS data plane, if the network were to use SR-IPv6 data plane, path size would be

increased even more because of the size of the IPv6 SID (16 bytes) in SRH.

(b). Apart from the TE case above, when deploying [I-D.ietf-mpls-sfc] or [I-D.xuclad-spring-sr-service-chaining], with the inclusion of services, or non-topological segments on the label stack, can also make the size of the stack much larger.

(c). Some SR-MPLS deployments need accounting statistics for path monitoring and traffic re-optimizations. [I-D.hegde-spring-traffic-accounting-for-sr-paths] and [I-D.cheng-spring-mpls-path-segment] propose solutions with various forms of path segments (either with special labels or PATH segment encoded at the bottom of the stack respectively). However, these proposals further increases the depth of SID stack, when it is compounded with MSD/RLDs of various nodes in the path.

Overall the additional path overhead in various SR deployments may cause the following issues:

a. HW Capabilities: Not all nodes in the path can support the ability to push or read label stack needed [I-D.ietf-isis-segment-routing-msd] to satisfy user/operator requirements, Alternate paths which meet these user/operator requirements may not be available.

b. Line Rate: Potential performance issues in deployments, which use SRH data plane with the increased size of the SRH with 16 byte SIDs.

c. MTU: Larger SID stacks on the data packet can cause potential MTU/fragmentation issues.

d. Header Tax: Some deployments, such as 5G, require minimal packet overhead in order to conserve network resources. Carrying 40 or 50 octets of data in a packet with hundreds of octet of header would be an unacceptable use of available bandwidth.

With the solution proposed in this document (Section 5) and Section 4), for Path-x in the example network Figure 1 above, SID stack would be reduced from 8 SIDs to a single SID.

1.3. Mitigation with MSD

The number of SIDs in the stack a node can impose is referred as Maximum SID Depth (MSD) capability [I-D.ietf-isis-segment-routing-msd], which must be taken into consideration when computing a path to transport a data packet in a

network implementing segment routing. [I-D.ietf-isis-mpls-elc] defines another MSD type, Readable Label Depth (RLD) that is used by a head-end to insert Entropy Label pair (ELI/EL) at appropriate depth, so it could be read by transit nodes. There are situations where the source routed path can be excessive as path represented by SR SIDs need to describe all the nodes and ELI/EL based on the readability of the nodes in that path.

MSDs (and RLD type) capabilities advertisement help mitigate the problem for a central entity to create the right source routed path per application/operator requirements. However the availability of actual paths meeting these requirements are still limited by the underlying hardware and their MSD capabilities in the data path.

2. Preferred Path Routing (PPR)

PPR mitigates the issues described in Section 1.2, while continuing to allow the direction of traffic along an engineered path through the network by replacing the label stack with a PPR-ID. The PPR-ID can either be a single label or a native destination address. To facilitate the use of a single label to describe an entire path, a new TLV is added to IS-IS, as described below in Section 3.

A PPR could be an SR path, a traffic engineered path computed based on some constraints, an explicitly provisioned Fast Re-Route (FRR) path or a service chained path. A PPR can be signaled by any node, computed by a central controller, or manually configured by an operator. PPR extends the source routing and path steering capabilities to native IP (IPv4 and IPv6) data planes without hardware upgrades; see Section 5.

2.1. PPR-ID and PPR Path Description

The PPR-ID describes a path through the network. For SR- MPLS this is an MPLS Label/SID and for SRv6 this is an IPv6-SID. For native IP data planes this is either IPv4 or IPv6 address/prefix.

The path identified by the PPR-ID is described as a set of Path Description Elements (PDEs), each of which represents a segment of the path. Each node determines location in the path as described, and forwards to the next segment/hop or label of the path description (see the Forwarding Procedure Example later in this document).

These PPR-PDEs as defined in Section 3.3, like SR SIDs, can represent topological elements like links/nodes, backup nodes, as well as non-topological elements such as a service, function, or context on a particular node. PPR-PDE optionally, can also have more information as described with in their Sub-TLVs.

A PPR path can be described as a Strict-PPR or a Loose-PPR. In a Strict-PPR all nodes/links on the path are described with SR SIDs for SR data planes or IPv4/IPV6 addresses for native IP data planes. In a Loose-PPR only some of the nodes/links from source to destination are described. More specifics and restrictions around Strict/Loose PPRs are described in respective data planes in Section 5. Each PDE is described as either an MPLS label towards the next hop in MPLS enabled networks, or as an IP next hop, in the case of either "plain"/"native" IP or SRv6 enabled networks. A PPR path is related to a set of PDEs using the following TLVs.

3. PPR Related TLVs

This section describes the encoding of PPR TLV. This TLV can be seen as having 4 logical section viz., encoding of the PPR-Prefix (IS-IS Prefix), encoding of PPR-ID, encoding of path description with an ordered PDE Sub-TLVs and a set of optional PPR attribute Sub-TLVs, which can be used to describe one or more parameters of the path. Multiple instances of this TLV MAY be advertised in IS-IS LSPs with different PPR-ID Type and with corresponding PDE Sub-TLVS. The PPR TLV has Type TBD (suggested value xxx), and has the following format:

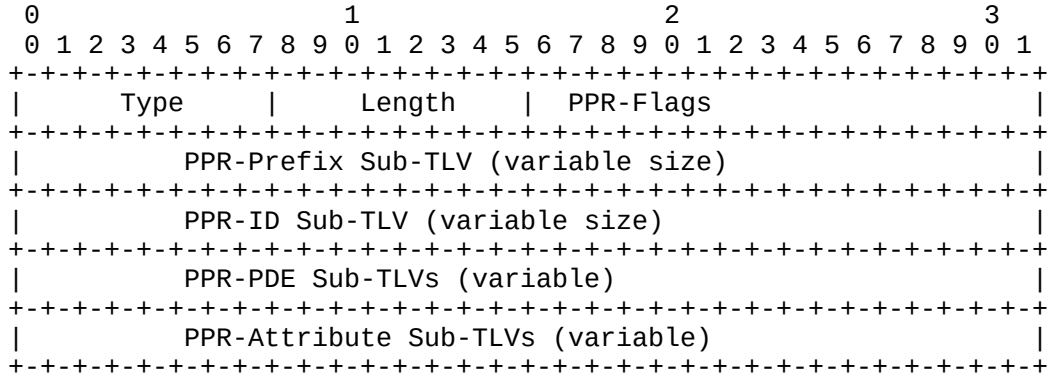


Figure 2: PPR TLV Format

- Type: TBD (IANA) from IS-IS top level TLV registry.
- Length: Total length of the value field in bytes.
- PPR-Flags: 2 Octet bit-field of flags for this TLV; described below.
- PPR-Prefix: A variable size sub-TLV representing the destination of the path being described. This is defined in Section 3.1.

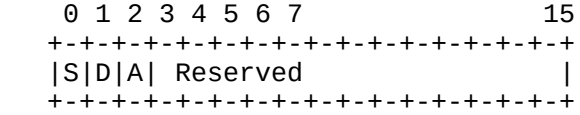
- o PPR-ID: A variable size Sub-TLV representing the data plane or forwarding identifier of the PPR. Defined in Section 3.2.

- o PPR-PDEs: Variable number of ordered PDE Sub-TLVs which represents the path. This is defined in Section 3.3.

- o PPR-Attributes: Variable number of PPR-Attribute Sub-TLVs which represent the path attributes. These are defined in Section 3.4.

The Flags field has the following flag bits defined:

PPR TLV Flags Format



1. S: If set, the PPR TLV MUST be flooded across the entire routing domain. If the S flag is not set, the PPR TLV MUST NOT be leaked between IS-IS levels. This bit MUST NOT be altered during the TLV leaking

2. D: When the PPR TLV is leaked from IS-IS level-2 to level-1, the D bit MUST be set. Otherwise, this bit MUST be clear. PPR TLVs with the D bit set MUST NOT be leaked from level-1 to level-2. This is to prevent TLV looping across levels.

3. A: The originator of the PPR TLV MUST set the A bit in order to signal that the prefixes and PPR-IDs advertised in the PPR TLV are directly connected to the originators. If this bit is not set, this allows any other node in the network advertise this TLV on behalf of the originating node of the PPR-Prefix. If PPR TLV is leaked to other areas/levels the A-flag MUST be cleared. In case if the originating node of the prefix must be disambiguated for any reason including, if it is a Multi Homed Prefix (MHP) or leaked to a different IS-IS level or because [RFC7794] X-Flag is set, then PPR-Attribute Sub-TLV Source Router ID SHOULD be included.

4. Reserved: For future use; MUST be set to 0 on transmit and ignored on receive.

The following sub-TLVs draw from a new registry for sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process.

3.1. PPR-Prefix Sub-TLV

The structure of PPR-Prefix is:

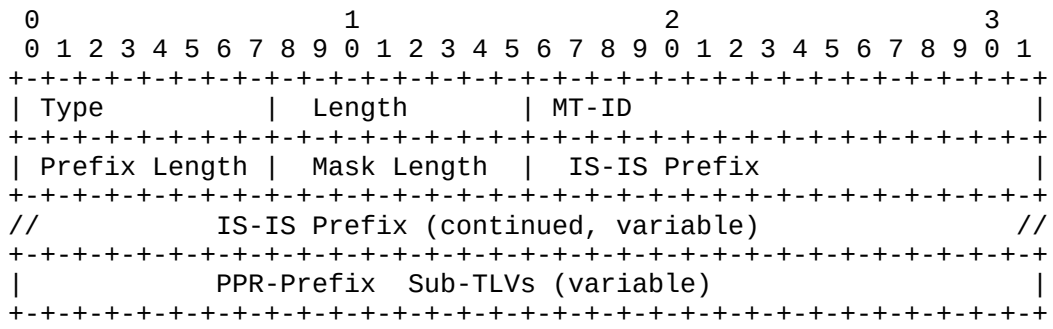


Figure 3: PPR-Prefix Sub-TLV Format

- Type: TBD (IANA to assign from sub-TLV registry described above).
- Length: Total length of the value field in bytes.
- MT-ID: The multi-topology identifier defined in [RFC5120]; the 4 most significant bits MUST be set to 0 on transmit and ignored on receive. The remaining 12-bit field contains the MT-ID.
- Prefix Length: The length of the prefix in bytes. For IPv4 it MUST be 4 and IPv6 it MUST be 16 bytes.
- Mask Length: The length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- IS-IS Prefix: The IS-IS prefix at the tail-end of the advertised PPR. This corresponds to a routable prefix of the originating node and it MAY have one of the [RFC7794] flags set (X-Flag/R-Flag/N-Flag). Value of this field MUST be 4 octets for IPv4 Prefix and MUST be 16 octets for IPv6 Prefix. Encoding is similar to TLV 135 [RFC5305] and TLV 236 [RFC5308] or MT-Capable [RFC5120] IPv4 (TLV 235) and IPv6 Prefixes (TLV 237) respectively.
- PPR-Prefix Sub-TLVs - TBD. These have 1 octet type, 1 octet length and value field is defined per the type field.

3.2. PPR-ID Sub-TLV

This is the actual data plane identifier in the packet header and could be of any data plane as defined in PPR-ID Type field. Both PPR-Prefix and PPR-ID MUST belong to a same node in the network.

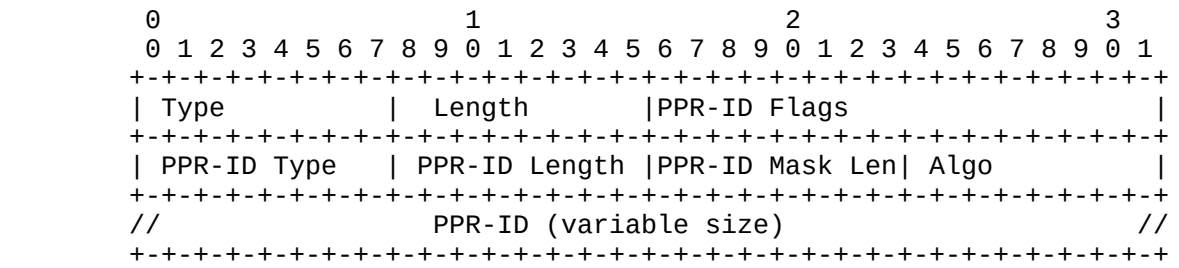
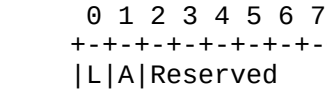


Figure 4: PPR-ID Sub-TLV Format

- Type: TBD (IANA to assign from sub-TLV registry described above).
- Length: Total length of the value field in bytes.
-
  - PPR-ID Flags: 2 Octet field for PPR-ID flags:
-

PPR-ID Flags Format



-

1. L: If set, the PPR path is a Loose-PPR. If the this flag is unset, then the PPR path is a Strict-PPR. A Strict-PPR lists every single node or adjacency in the path description from source to the destination.
2. A: If set, all non-PPR path nodes in the IS-IS area/domain MUST add a FIB entry for the PPR-ID with NH set to the shortest path NH for the prefix being advertised. The use of this is TBD. By default this flag MUST be unset.
3. Reserved: For future use; MUST be set to 0 on transmit and ignored on receive.

-

  * PPR-ID Type: Data plane type of PPR-ID. This is a new registry (TBD IANA - Suggested values as below) for this Sub-TLV and the defined types are as follows:

o

A. Type: 1 MPLS SID/Label

B. Type: 2 Native IPv4 Address/Prefix

C. Type: 3 Native IPv6 Address/Prefix

D. Type: 4 IPv6 SID in SRv6 with SRH

- PPR-ID Length: Length of the PPR-ID field in octets and this depends on the PPR-ID type. See PPR-ID below for the length of this field and other considerations.

- PPR-ID Mask Length: It is applicable for only for PPR-ID Type 2, 3 and 4. For Type 1 this value MUST be set to zero. It contains the length of the PPR-ID Prefix in bits. Only the most significant octets of the Prefix are encoded. This is needed, if PPR-ID followed is an IPv4/IPv6 Prefix instead of 4/16 octet Address respectively.

- Algo: 1 octet value represents the SPF algorithm. Algorithm registry is as defined in [I-D.ietf-isis-segment-routing-extensions].

- PPR-ID: This is the Preferred Path forwarding identifier that would be on the data packet. The value of this field is variable and it depends on the PPR-ID Type - for Type 1, this is and MPLS SID/Label. For Type 2 this is a 4 byte IPv4 address. For Type 3 this is a 16 byte IPv6 address. For Type 2 and Type 3 encoding is similar to "IS-IS Prefix" as specified in Section 3.1. For Type 4, it is a 16 byte IPv6 SID.

For PPR-ID Type 2, 3 or 4, if the PPR-ID Len is set to non-zero value, then the PPR-ID MUST NOT be advertised as a routable prefix in TLV 135, TLV 235, TLV 236 and TLV 237. Also PPR-ID MUST belong to the node where Prefix is advertised from. PPR-ID Len = 0 case is a special case and is discussed in Section 4.1.

3.3. PPR-PDE Sub-TLV

This Sub-TLV represents the PPR Path Description Element (PDE). PPR-PDEs are used to describe the path in the form of set of contiguous and ordered Sub-TLVs, where first Sub-TLV represents (the top of the

stack in MPLS data plane or) first node/segment of the path. These set of ordered Sub-TLVs can have both topological elements and non-topological elements (e.g., service segments).

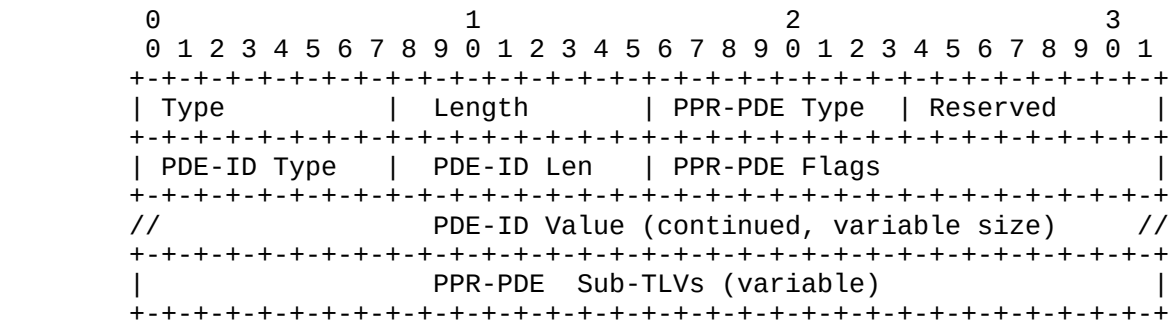


Figure 5: PPR-PDE Sub-TLV Format

- Type: TBD (See IANA for suggested value) from IS-IS PPR TLV Section 3 Sub-TLV registry.
- Length: Total length of the value field in bytes.
- PPR-PDE Type: A new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 Topological

b. Type: 2 Non-Topological

- PDE-ID Type: 1 Octet PDE-forwarding IDentifier Type. A new registry (TBD IANA) for this Sub-TLV and the defined types and corresponding PDE-ID Len, PDE-ID Value are as follows:

a. Type 1: SID/Label type as defined in [I-D.ietf-isis-segment-routing-extensions]. PDE-ID Len and PDE-ID Value fields are per Section 2.3 of the referenced document.

b. Type 2: SR-MPLS Prefix SID. PDE-ID Len and PDE-ID Value are same as Type 1.

c. Type 3: SR-MPLS Adjacency SID. PDE-ID Len and PDE-ID Value are same as Type 1.

d. Type 4: IPv4 Address. PDE-ID Len is 4 bytes and PDE-ID Value is 4 bytes IPv4 address encoded similar to IPv4 Prefix described in Section 3.1.

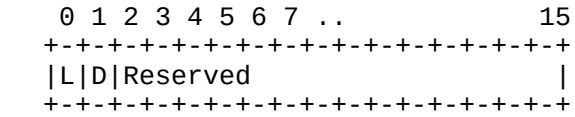
e. Type 5: IPv6 Address. PDE-ID Len is 16 bytes and PDE-ID Value is 16 bytes IPv6 address encoded similar to IPv6 Prefix described in Section 3.1.

f. Type 6: SRv6 Node SID as defined in [I-D.bashandy-isis-srv6-extensions]. PDE-ID Len and PDE-ID Value are as defined in SRv6 SID from the refrenced draft.

g. Type 7: SRv6 Adjacency-SID. PDE-ID Len and PDE-ID Values are similar to SRv6 Node SID above.

o PPR-PDE Flags: 2 Octet bit-field of flags; described below:

PPR-PDE Flags Format



1. L: Loose Bit. Indicates the type of next "Topological PDE-ID" in the path description and overrides the L bit in Section 3.2. If set, the next PDE is Loose. If this flag is unset, the next Topological PDE is Strict Type.

2. D: Destination bit. By default this bit MUST be unset. This bit MUST be set only for PPR-PDE Type is 1 i.e., Topological and this PDE represents the node PPR-Prefix Section 3.1 belongs to, if there is no sub-sub-TLV to override PPR-Prefix and PPR-ID values.

3. Reserved: 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

o PPR-PDE Sub-TLVs: TBD. These have 1 octet type, 1 octet length and value field is defined per the type field.

3.4. PPR-Attributes Sub-TLV

PPR-Attribute Sub-TLVs describe the attributes of the path. The following sub-TLVs draw from a new registry for sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process:

o Type 1 (Suggested Value - IANA TBD): Packet Traffic accounting Sub-TLV. Length 0 and no value field. Specifies to create a counter to count number of packets forwarded on this PPR-ID on each node in the path description.

- Type 2 (Suggested Value - IANA TBD): Traffic statistics in Bytes Sub-TLV. Length 0 and no value field. Specifies to create a counter to count number of bytes forwarded on this PPR-ID specified in the network header (e.g. IPv4, IPv6) on each node in the path description.

- Type 3 (Suggested Value - IANA TBD): PPR-Prefix originating node's IPv4 Router ID Sub-TLV. Length and Value field are as specified in [RFC7794].

- Type 4 (Suggested Value - IANA TBD): PPR-Prefix originating node's IPv6 Router ID Sub-TLV. Length and Value field are as specified in [RFC7794].

- Type 5 (Suggested Value - IANA TBD): PPR-Metric Sub-TLV. Length 4 bytes, and Value is metric of this path represented through the PPR-ID. Different nodes can advertise the same PPR-ID for the same Prefix with a different set of PPR-PDE Sub-TLVs and the receiving node MUST consider the lowest metric value (TBD more, on what happens when metric is same for two different set of PPR-PDE Sub-TLVs).

4. PPR Processing Procedure Example

As specified in Section 2, a PPR can be a TE path, locally provisioned by the operator or by a controller. Consider the following IS-IS network to describe the operation of PPR TLV as defined in Section 3:

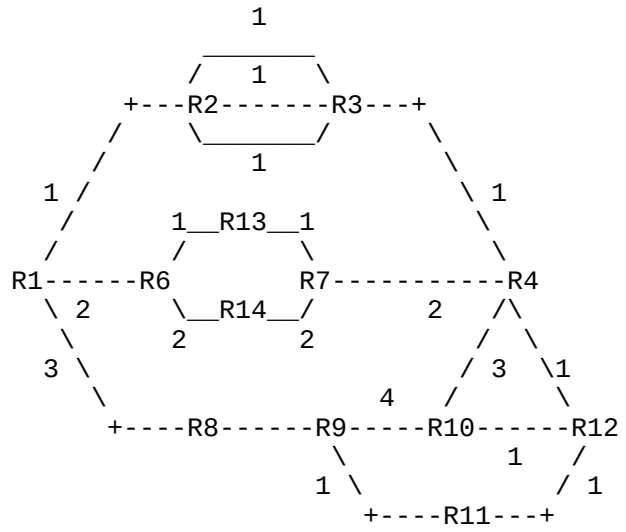


Figure 6: IS-IS Network

In the (Figure 6) shown, consider node R1 as an ingress node, or a head-end node, and the node R4 MAY be an egress node or another head-end node. The numbers shown on links between nodes R1-R14 indicate the bi-directional IS-IS metric as provisioned. R1 may be configured to receive TE source routed path information from a central entity (PCE [RFC5440], Netconf [RFC6241] or a Controller) that comprises of PPR information which relates to sources that are attached to R1. It is also possible to have a PPR provisioned locally by the operator for non-TE needs (FRR or for chaining certain services).

The PPR TLV (as specified in Section 3) is encoded as an ordered list of PPR-PDEs from source to a destination node in the network and is represented with a PPR-ID Section 3.2. The PPR TLV includes PPR-PDE Sub-TLVS Section 3.3, which represent both topological and non-topological elements and specifies the actual path towards a PPR-Prefix at R4.

- The shortest path towards R4 from R1 are through the following sequence of nodes: R1-R2-R3-R4 based on the provisioned metrics.

- The central entity can define a few PPRs from R1 to R4 that deviate from the shortest path based on other network characteristic requirements as requested by an application or service. For example, the network characteristics or performance requirements may include bandwidth, jitter, latency, throughput, error rate, etc.

- o A first PPR may be identified by PPR-ID = 1 (value) and may include the path of R1-R6-R7-R4 for a Prefix advertised by R4. This is an example for a Loose-PPR and 'L' bit MUST be set on Section 3.2.

- o A second PPR may be identified by PPR-ID = 2 (value) and may include the path of R1-R8-R9-R10-R4. This is an example for a Strict-PPR and 'L' bit MUST be unset on Section 3.2. Though this example shows PPR with all nodal SIDs, it is possible to have a PPR with combination of node and adjacency SIDs (local or global) or with PPR-PDE Type set to Non-Topological as defined in Section 3.3 elements along with these.

4.1. PPR TLV Processing

The first topological sub-object or PDE (Section 3.3) relative to the beginning of PPR Path contains the information about the first node (e.g. in SR-MPLS it's the topmost label). The last topological sub-object or PDE contains information about the last node (e.g. in SR-MPLS it's the bottommost label).

Each receiving node, determine whether an advertised PPR includes information regarding the receiving node. Before processing any further, validation MUST be done to see if any PPR topological PDE is seen more than once (possible loop), if yes, this PPR TLV MUST be ignored. Processing of PPR TLVs can be done, during the end of the SPF computation (for MTID that is advertised in this TLV) and for the each prefix described through PPR TLV. For example, node R9 receives the PPR information, and ignores the PPR-ID=1 (Section 4) because this PPR TLV does not include node R9 in the path description/ordered PPR-PDE list.

However, node R9 may determine that the second PPR identified by PPR-ID = 2 does include the node R9 in its PDE list. Therefore, node R9 updates the local forwarding database to include an entry for the destination address of R4 indicates, that when a data packet comprising a PPR-ID of 2 is received, forward the data packet to node R10 instead of R11. This is even though from R9 the shortest path to reach R4 via R11 (Cost 3: R9-R11-R12-R4) it chooses the nexthop to R10 to reach R4 as specified in the PPR path description. Same process happens to all nodes or all topological PDEs as described in the PPR TLV.

In summary, the receiving node checks first, if this node is on the path by checking the node's topological elements (with PPR-PDE Type set to Topological) in the path list. If yes, it adds/adjusts the shortest path nexthop computed towards PPR Prefix to the shortest path nexthop towards the next topological PDE in the PPR's Path.

For PPR-ID (Section 3.2) Type 2, 3 or 4, if the PPR-ID Len is set to 0, then Prefix would also become the PPR-ID (a special case). This can be used for some situations, where certain optimizations are required in the network. For this, path described in the PPR TLV SHOULD be completely dis-joint from the shortest path route to the prefix. If the disjoint-ness property is not maintained then the traffic MAY not be using the PPR path, as and when it encounters any node which is not in the path description.

5. PPR Data Plane aspects

Data plane for PPR-ID is selected by the entity (e.g., a controller, locally provisioned by operator), which selects a particular PPR in the network. Section 3.2 defines various data plane identifier types and a corresponding data plane identifier is selected by the entity which selects the PPR. Other data planes other than described below can also use this TLV to describe the PPR. Further details TBD.

5.1. SR-MPLS with PPR

If PPR-ID Type is 1, then the PPR belongs to SR-MPLS data plane and the complete PPR stack is represented with a unique SR SID/Label and this gets programmed on the data plane of each node, with the appropriate nexthop computed as specified in Section 4. PPR-ID here is a label/index from the SRGB (like another node SID or global-ADJ SID). PPR path description here is a set of ordered SIDs represented with PPR-PDE (Section 3.2) Sub-TLVs. Non-Topological segments also programmed in the forwarding to enable specific function/service, when the data packet hits with corresponding PPR-ID.

Based on 'L' flag in PPR-ID Flags (Section 3.2), for SR-MPLS data plane either 1 label or 2 labels need to be provisioned on individual nodes on the path description. For the example network in Section 4, for PPR-ID=1, which is a loose path, node R6 programs the bottom label as PPR-ID and the top label as the next topological PPR-PDE in the path, which is a node SID of R7. The NH computed at R6 would be the shortest path towards R7 i.e., the interface towards R13. If 'L' flag is unset only PPR-ID is programmed on the data plane with NH set to the shortest path towards next topological PPR-PDE.

5.2. SRv6 with PPR

If PPR-ID Type is 4, the PPR belongs to SRv6 with SRH data plane and the complete PPR stack is represented with IPv6 SIDs and this gets programmed on the data plane with the appropriate nexthop computed as specified in Section 4. PPR-ID here is a SRv6 SID. PPR path description here is a set of ordered SID TLVs similar to as specified in Section 5.1. One way PPR-ID would be used in this case is by

setting the same as the destination IPv6 address and SL field in SRH would be set to 0; however SRH [I-D.ietf-6man-segment-routing-header] can contain any other TLVs and non-topological SIDs as needed.

5.3. PPR Native IP Data Planes

If PPR-ID Type is 2 then source routing and packet steering can be done in IPv4 data plane (PPR-IPv4), along the path as described in PPR Path description. This is achieved by setting the destination IP address as PPR-ID, which is an IPv4 address in the data packet (tunneled/encapsulated). There is no data plane change or upgrade needed to support this. However this is applicable to only Strict-PPR paths ('L' bit as specified in Section 3.2 MUST be unset).

Similarly for PPR-ID Type is 3, then source routing and packet steering can be done in IPv6 data plane (PPR-IPv6), along the path as described in PPR Path description. Whatever specified above for IPv4 applies here too, except that destination IP address of the data packet is IPv6 Address (PPR-ID). This doesn't require any IPv6 extension headers (EH), if there is no metadata/TLVs need to be carried in the data packet.

6. PPR Scaling Considerations

In a network of N nodes O(N^2) total (unidirectional) paths are necessary to establish any-to-any connectivity, and multiple (k) such path sets may be desirable if multiple path policies are to be supported (lowest latency, highest throughput etc.).

In many solutions and topologies, N may be small enough and/or only a small set of paths need to be preferred paths, for example for high value traffic (DetNet, some of the defined 5G slices), and then the technology specified in this document can support these deployments.

However, to address the scale needed when a larger number of PPR paths are required, the PPR TREE structure can be used [I-D.draft-ce-ppr-graph-00]. Each PPR Tree uses one label/SID and defines paths from any set of nodes to one destination, thus reduces the number of entries needed in SRGB at each node (for SR-MPLS data plane Section 5.1). These paths form a tree rooted in the destination. In other word, PPR Tree identifiers are destination identifiers and PPR Treed are path engineered destination routes (like IP routes) and it scaling simplifies to linear in N i.e., O(k*N).

7. PPR Traffic Accounting

Section 3.4 defines few PPR-Attributes to indicate creation of traffic accounting statistics in each node of the PPR path description. Presence of "Packet Traffic Accounting" and "Traffic Statistics" Sub-TLVs instruct to provision the hardware, to account for the respective traffic statistics. Traffic accounting should happen, when the actual data traffic hits for the PPR-ID in the forwarding plane. This allows more granular and dynamic enablement of traffic statistics for only certain PPRs as needed.

This approach, thus is more safe and secure than any mechanism that involves creation of the state in the nodes with the data traffic itself. This is because, creation and deletion of the traffic accounting state for PPRs happen through IS-IS LSP processing and IS-IS protocol control plane security Section 10 options are applicable to this TLV.

How the traffic accounting is distributed to a central entity is out of scope of this document. One can use any method (e.g. gRPC) to extract the PPR-ID traffic stats from various nodes along the path.

8. Acknowledgements

Thanks to Alex Clemm, Lin Han, Toerless Eckert and Kiran Makhijani for initial discussions on this topic. Thanks to Kevin Smith and Stephen Johnson for various deployment scenarios applicability from ETSI WGs perspective. Authors also acknowledge Alexander Vainshtein for detailed discussions and suggestions on this topic.

Earlier versions of draft-ietf-isis-segment-routing-extensions have a mechanism to advertise EROs through Binding SID.

9. IANA Considerations

This document requests the following new TLV in IANA IS-IS TLV code-point registry.

TLV #	Name
TBD	PPR TLV

This document requests IANA to create a new Sub-TLV registry for PPR TLV Section 3 with the following initial entries (suggested values):

Sub-TLV #	Sub-TLV Name

1	PPR-Prefix (Section 3.1)
2	PPR-ID (Section 3.2)
3	PPR-PDE (Section 3.3)

This document also requests IANA to create a new Sub-TLV registry for PPR Path attributes with the following initial entries (suggested values):

Sub-TLV #	Sub-TLV Name
1	Packet Traffic Accounting (Section 3.4)
2	Traffic Statistics (Section 3.4)
3	PPR-Prefix Source IPv4 Router ID (Section 3.4)
4	PPR-Prefix Source IPv6 Router ID (Section 3.4)
5	PPR-Metric (Section 3.4)

This document requests additional IANA registries in an IANA managed registry "Interior Gateway Protocol (IGP) Parameters" for various PPR TLV parameters. The registration procedure is based on the "Expert Review" as defined in [RFC8126]. The suggested registry names are:

- "PPR-Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3 of this document.
- "PPR-Flags" - 1 Octet. Bits as described in Section 3 of this document.
- "PPR-ID Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.2 of this document.
- "PPR-ID Flags" - 1 Octet. Bits as described in Section 3.2 of this document.
- "PPR-PDE Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.3 of this document.
- "PPR-PDE Flags" - 1 Octet. Bits as described in Section 3.3 of this document.
- "PDE-ID Type" - Types are an unsigned 8 bit numbers. Values are as defined in Section 3.3 of this document.

10. Security Considerations

Security concerns for IS-IS are addressed in [RFC5304] and [RFC5310]. Further security analysis for IS-IS protocol is done in [RFC7645] with detailed analysis of various security threats and why [RFC5304] should not be used in the deployments. Advertisement of the additional information defined in this document introduces no new security concerns in IS-IS protocol. However as this extension is related to SR-MPLS and SRH data planes as defined in [I-D.ietf-spring-segment-routing], those particular data plane security considerations does apply here.

11. References

11.1. Normative References

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-12 (work in progress), May 2018.

[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.

[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>.

11.2. Informative References

[I-D.bashandy-isis-srv6-extensions]
Ginsberg, L., Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-03 (work in progress), June 2018.

[I-D.cheng-spring-mpls-path-segment]
Cheng, W., Wang, L., Li, H., Chen, M., Zigler, R., and S. Zhan, "Path Segment in MPLS Based Sement Routing Network", draft-cheng-spring-mpls-path-segment-01 (work in progress), March 2018.

[I-D.hegde-spring-traffic-accounting-for-sr-paths]
Hegde, S., "Traffic Accounting for MPLS Segment Routing Paths", draft-hegde-spring-traffic-accounting-for-sr-paths-01 (work in progress), October 2017.

[I-D.ietf-6man-segment-routing-header]
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-14 (work in progress), June 2018.

[I-D.ietf-isis-mpls-elc]
Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability and Readable Label-stack Depth Using IS-IS", draft-ietf-isis-mpls-elc-03 (work in progress), January 2018.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-18 (work in progress), June 2018.

[I-D.ietf-mpls-sfc]
Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", draft-ietf-mpls-sfc-01 (work in progress), May 2018.

[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.

[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>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[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>.

[RFC7645] Chunduri, U., Tian, A., and W. Lu, "The Keying and Authentication for Routing Protocol (KARP) IS-IS Security Analysis", RFC 7645, DOI 10.17487/RFC7645, September 2015, <https://www.rfc-editor.org/info/rfc7645>.

[RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <https://www.rfc-editor.org/info/rfc7794>.

[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>.

[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>.

Authors' Addresses

Uma Chunduri
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: uma.chunduri@huawei.com

Richard Li
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: renwei.li@huawei.com

Russ White
LinkedIn
Oak Island, NC 28465
USA

Email: russ@riw.us

Jeff Tantsura
Nuage Networks
755 Ravendale Drive
Mountain View, CA 94043
USA

Email: jefftant.ietf@gmail.com

Luis M. Contreras
Telefonica
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

Yingzhen Qu
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yingzhen.qu@huawei.com

Chunduri, et al. Expires January 3, 2019 [Page 25]

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

U. Chunduri
Y. Qu
Huawei USA
R. White
LinkedIn
J. Tantsura
Nuage Networks
L. Contreras
Telefonica
July 2, 2018

Preferred Path Routing (PPR) in OSPF
draft-chunduri-lsr-ospf-preferred-path-routing-01

Abstract

This document specifies a Preferred Path Routing (PPR) mechanism to simplify the path description of data plane traffic in Segment Routing (SR) deployments with OSPFv2 and OSPFv3 protocols. PPR aims to mitigate the MTU and data plane processing issues that may result from SR packet overheads; and also supports traffic measurement, accounting statistics and further attribute extensions along the paths. Preferred Path Routing is achieved through the addition of descriptions to OSPF advertised prefixes, and mapping those to a PPR data-plane identifier.

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 [RFC2119], RFC8174 [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 January 3, 2019.

Table of Contents

1. Introduction

In a network implementing Segment Routing (SR), packets are steered through the network using Segment Identifiers (SIDs) carried in the packet header. Each SID uniquely identifies a segment as defined in [I-D.ietf-spring-segment-routing]. SR capabilities are defined for MPLS and IPv6 data planes called SR-MPLS and SRv6 respectively.

In SR-MPLS, a segment is encoded as a label and an ordered list of segments is encoded as a stack of labels on the data packet. In SRv6, a segment is encoded as an IPv6 address, with in a new type of IPv6 hop-by-hop routing header/extension header (EH) called SRH [I-D.ietf-6man-segment-routing-header], where an ordered list of IPv6 addresses/segments is encoded in SRH.

The issues caused by the large SID depth, and existing methods for mitigation are introduced in [I-D.chunduri-lsr-isis-preferred-path-routing] section 1.2 and 1.3. To mitigate these issues , and also to facilitate forwarding plane a mechanism to identify the path with a corresponding data plane identifier for accounting of traffic for SR paths, this draft proposes a new OSPFv2 PPR TLV (Section 2), OSPFv3 PPR TLV (Section 3) to use the path with a corresponding data plane identifier.

Preferred Path Routing means enabling route computation based on the specific path described along with the prefix as opposed to shortest path towards the prefix. This also further described in Section 2 of [I-D.chunduri-lsr-isis-preferred-path-routing].

Any prefix advertised with a path description from any node in the network is called Preferred Path Route. A PPR could be an SR path, an explicitly provisioned Fast Re-Route (FRR) path or a service chained path. A PPR can be signaled by any node, which receives the SR path computed by a central controller, or by operator by statically configuring the same on a node in the network.

With corresponding data plane, Section 4 mechanism as in [I-D.chunduri-isis-preferred-path-routing], reduces the SID stack in the data plane with a single PPR ID.

1.1. Acronyms

EL - Entropy Label

ELI - Entropy Label Indicator

MPLS - Multi Protocol Label Switching

MSD - Maximum SID Depth

MTU - Maximum Transferrable Unit

NSP - Non Shortest Path

SID - Segment Identifier

SPF - Shortest Path First

SR - Segment Routing

SRH - Segment Routing Header

SR-MPLS - Segment Routing with MPLS data plane

SRv6 - Segment Routing with Ipv6 data plane with SRH

SRH - IPv6 Segment Routing Header

TE - Traffic Engineering

2. OSPFv2 PPR TLV

Extended Prefix Opaque LSAs defined in [RFC7684] are used for advertisements of PPRs. This section describes the encoding of PPR TLV. This TLV can be seen as having 4 logical section viz., encoding of the OSPFv2 Prefix, encoding of PPR-ID, encoding of path description with an ordered PDE Sub-TLVs and a set of optional PPR attribute Sub-TLVs, which can be used to describe one or more parameters of the path. Multiple OSPF PPR TLVs MAY be advertised in each OSPF Extended Prefix Opaque LSA, but all TLVs included in a single OSPF Extended Prefix Opaque LSA MUST have the same flooding scope.

The PPR TLV has Type TBD (suggested value xxx), and has the following format:

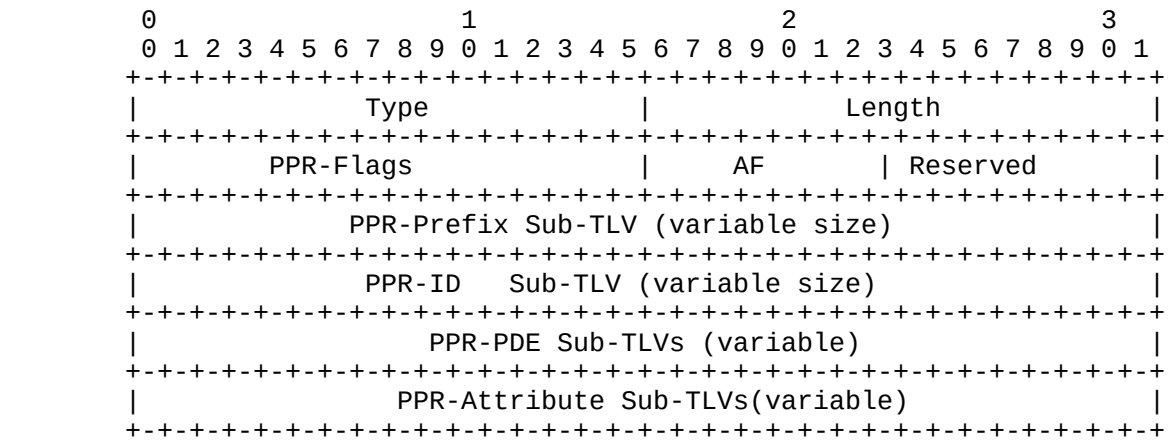
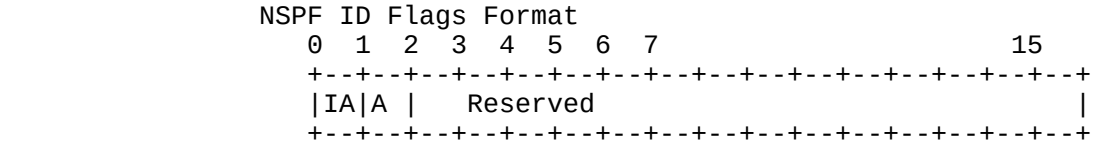


Figure 1: OSPFV2 PPR TLV Format

- Type - TBD (IANA)from OSPF Extended Prefix Opaque LSA registry.
- Length - Total length of the value field in bytes (variable).
- PPR-Flags - 2 Octet flags for this TLV are described below.
- AF - Address family for the prefix. Currently, the only supported value is 0 for IPv4 unicast. The inclusion of address family in this TLV allows for future extension.
- Reserved - 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.
- PPR-Prefix - This is a variable size Sub-TLV, which represents the prefix for which path description is being attached to. This is defined in Section 2.2.
- PPR-ID - This is a variable size Sub-TLV, which represents the data plane or forwarding identifier of the PPR. This is defined in Section 2.3.
- PPR-PDEs - Variable number of ordered PDE Sub-TLVs which represents the path. This is defined in Section 2.4.
- PPR-Attributes - Variable number of PPR-Attribute Sub-TLVs which represent the path attributes. These are defined in Section 2.5.

2.1. PPR-Flags

Flags: 2 octet field of PPR TLV has following flags defined:



w=Where:

IA-Flag: Inter-Area flag. If set, advertisement is of inter-area type. An Area Boarder Router (ABR) that is advertising the OSPF PPR TLV between areas MUST set this bit.

A: The originator of the PPR TLV MUST set the A bit in order to signal that the prefixes and PPR-IDs advertised in the PPR TLV are directly connected to the originators. If this bit is not set, this allows any other node in the network advertise this TLV on behalf of the originating node of the "OSPF Prefix". If PPR TLV is propagated to other areas the A-flag MUST be cleared. In case if the originating node of the prefix has to be disambiguated for any reason including, if it is a Multi Homed Prefix (MHP) or propagated to a different OSPF area, then PPR-Attribute Sub-TLV Source Router ID SHOULD be included.

Reserved: Reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

2.2. PPR-Prefix Sub-TLV

The structure of PPR-Prefix, for which path description is attached to is as follows:

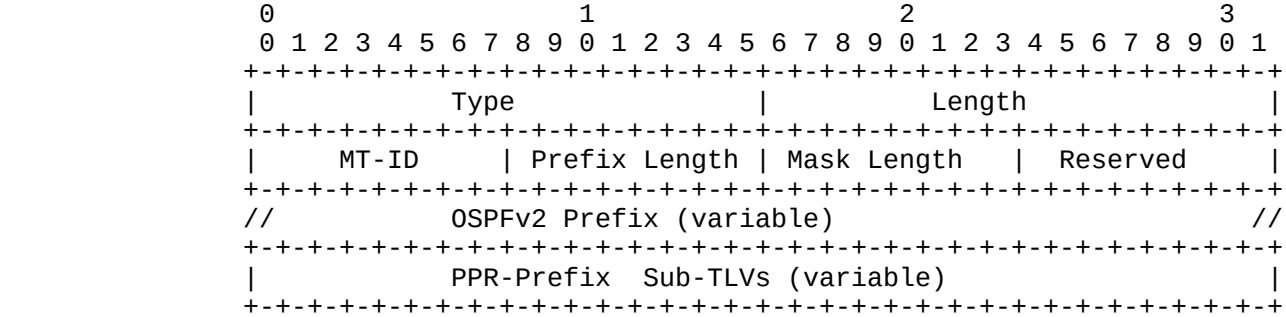


Figure 2: PPR-Prefix Sub-TLV Format

- Type - TBD (See IANA for suggested value) from OSPFv2 PPR TLV Section 2 Sub-TLV registry.
- Length - Total length of the value field in bytes (variable).
- MT-ID - Multi-Topology ID (as defined in [RFC4915]).
- Prefix Len - contains the length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- Mask Length - The length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- OSPFv2 Prefix - represents the OSPFv2 prefix at the tail-end of the advertised PPR. For the address family IPv4 unicast, the prefix itself is encoded as a 32-bit value. The default route is represented by a prefix of length 0.
- PPR-Prefix Sub-TLVs - TBD. It has 2 octet type, 2 octet length and value field is defined per type.

2.3. PPR-ID Sub-TLV

This represents the actual data plane identifier in the packet and could be of any data plane as defined in PPR-ID-type field. Both OSPF Prefix and PPR-ID MUST belong to a same node in the network.

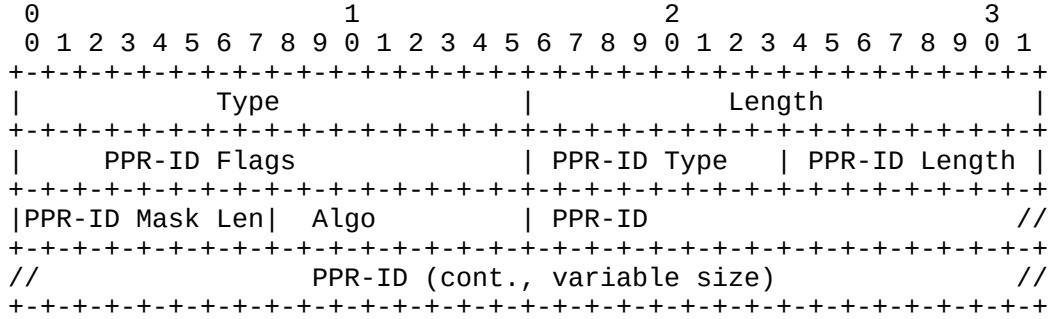


Figure 3: PPR-ID Sub-TLV Format

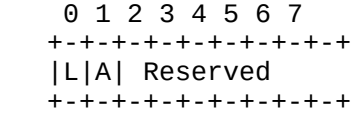
- Type - TBD (See IANA for suggested value) from OSPFv2 PPR TLV Section 2 Sub-TLV registry.
- Length - Total length of the value field in bytes (variable).
- PPR-ID Type - Data plane type of PPR-ID. This is a new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 MPLS SID/Label

b. Type: 2 Native IPv4 Address/Prefix

o PPR-ID Flags - 2 Octet field for PPR-ID flags:

PPR-ID Flags Format



1. L - If set, the PPR path is a Loose-PPR. If the flag is unset, then the path described is a Strict-PPR. A Strict-PPR lists every single node or adjacency in the path description from source to the destination.

2. A - If set, all non-PPR path nodes in the OSPF area MUST add a FIB entry for the PPR-ID with NH set to the shortest path NH for the prefix being advertised. The use of this is TBD. By default this MUST be unset.

3. Reserved - Reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

o PPR-ID Length - Length of the PPR-ID field in octets and this depends on the PPR-ID type. See PPR-ID below for the length of this field and other considerations.

o PPR-ID Mask Len - It is applicable for only for PPR-ID Type 2. For Type 1 this value MUST be set to zero. It contains the length of the PPR-ID Prefix in bits. Only the most significant octets of the Prefix are encoded. This is needed, if PPR-ID followed is an IPv4 Prefix instead of 4 octet Address respectively.

o Algo - 1 octet value represents the SPF algorithm. Algorithm registry is as defined in [I-D.ietf-ospf-segment-routing-extensions].

o PPR-ID - This is the Preferred Path forwarding identifier that would be on the data packet. The value of this field is variable and it depends on the PPR-ID Type - for Type 1, this is and MPLS SID/Label. For Type 2 this is a 4 byte IPv4 address.

2.4. PPR-PDE Sub-TLV

This is a new Sub-TLV type in PPR TLV Section 2 and is called as PPR Path Description Element (PDE). PPR-PDEs are used to describe the path in the form of set of contiguous and ordered Sub-TLVs, where first Sub-TLV represents (the top of the stack in MPLS data plane or) first node/segment of the path. These set of ordered Sub-TLVs can have both topological SIDs and non-topological SIDs (e.g., service segments).

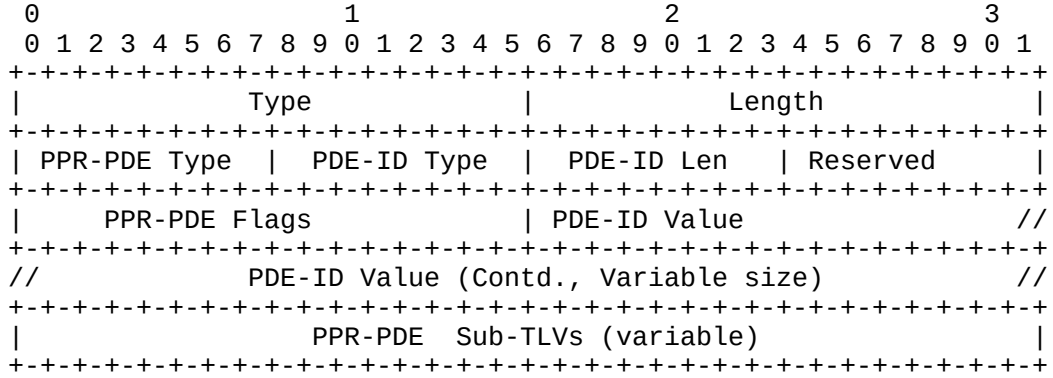


Figure 4: PPR-PDE Sub-TLV Format

- Type - TBD (See IANA for suggested value) from OSPFv2 PPR TLV Section 2 Sub-TLV registry.
- Length - Total length of the value field in bytes (variable).
- PPR-PDE Type - This is a new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 Topological

b. Type: 2 Non-Topological

- PDE-ID Type - 1 Octet PDE-forwarding IDentifier Type. This is a new registry (TBD IANA) for this Sub-TLV and the defined types and corresponding PDE-ID Len, PDE-ID Value are as follows:

a. Type 1: SID/Label Sub-TLV as defined in [I-D.ietf-ospf-segment-routing-extensions]. PDE-ID Len and PDE-ID Value fields are per Section 2.1 of the referenced document.

b. Type 2: SR-MPLS Prefix SID. PDE-ID Len and PDE-ID Value are same as Type 1.

c. Type 3: SR-MPLS Adjacency SID. PDE-ID Len and PDE-ID Value are same as Type 1.

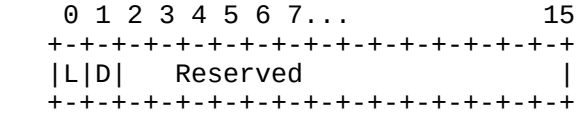
d. Type 4: IPv4 Address. PDE-ID Len is 4 bytes and PDE-ID Value is 4 bytes IPv4 address encoded similar to IPv4 Prefix described in Section 2.2.

- PDE-ID Len - 1 Octet. Length of PDE-ID field.

- Reserved - 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

- PPR-PDE Flags - 2 Octet flags for this TLV are described below:

PPR-PDE Flags Format



1. L: This bit indicates the type of next "Topological PDE-ID" in the path description and overrides the L bit in Section 2.3. If set, the next PDE is Loose. If this flag is unset, the next Topological PDE is Strict Type.

2. D: By default this bit MUST be unset. This bit MUST be set only for PPR-PDE Type is Topological and this PDE represents the PDE-ID corresponding to the PPR-Prefix Section 2.2.

3. Reserved: Reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

- PPR-PDE Sub-TLVs - TBD. It has 2 octet type, 2 octet length and value field is defined per type.

2.5. PPR-Attributes Sub-TLV

PPR-Attribute Sub-TLVs describe the attributes of the path. The following Sub-TLVs draw from a new registry for Sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process:

- Type 1 (Suggested Value - IANA TBD): This is Packet Traffic accounting Sub-TLV. Length 0 No value field. Specifies to create a counter to count number of packets forwarded on this PPR-ID on each node in the path description.

- Type 2 (Suggested Value - IANA TBD): This is Traffic statistics in Bytes Sub-TLV. Length 0 No value field. Specifies to create a counter to count number of bytes forwarded on this PPR-ID specified in the network header (e.g. IPv4, IPv6) on each node in the path description.

- Type 3 (Suggested Value - IANA TBD): PPR-Metric Sub-TLV. Length 4 bytes, and Value is metric of this path represented through the PPR-ID. Different nodes can advertise the same PPR-ID for the same Prefix with a different set of PPR-PDE Sub-TLVs and the receiving node MUST consider the lowest metric value (TBD more, on what happens when metric is same for two different set of PPR-PDE Sub-TLVs).

3. OSPFv3 PPR TLV

The OSPFv3 PPR TLV s a top level TLV of the following LSAs defined in [I-D.ietf-ospf-ospfv3-lsa-extend].

E-Intra-Area-Prefix-LSA

E-Inter-Area-Prefix-LSA

E-AS-External-LSA

E-Type-7-LSA

Multiple OSPFv3 PPR TLVs MAY be advertised in each LSA mentioned above. The OSPFv3 PPR TLV has the following format:

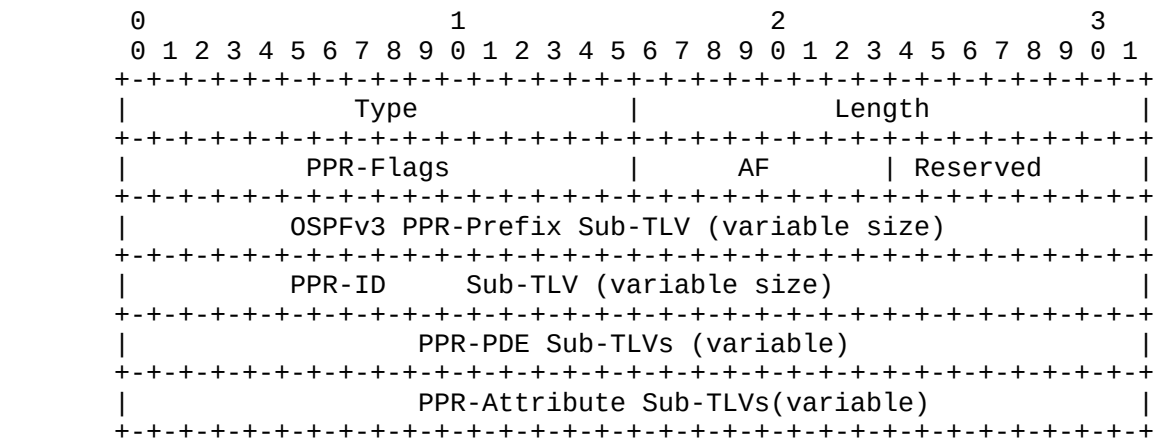


Figure 5: OSPFv3 PPR TLV Format

- Type - TBD (IANA)from OSPF Extended Prefix Opaque LSA registry.
- Length - Total length of the value field in bytes (variable).
- PPR-Flags - 2 Octet flags for this TLV are described below.
- AF: Address family for the prefix.
-

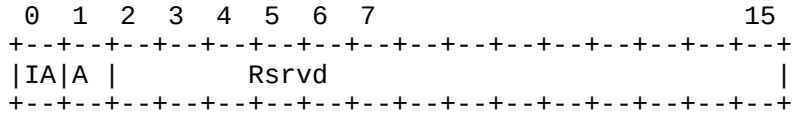
  AF: 0 - IPv4 unicast

  AF: 1 - IPv6 unicast

- Reserved - 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

Flags: 2 octet field. The following flags are defined:

OSPFv3 PPR Flags Format



IA-Flag: Inter-Area flag. If set, advertisement is of inter-area type. An ABR that is advertising the OSPF PPR TLV between areas MUST set this bit.
[I-D.ietf-ospf-ospfv3-segment-routing-extensions]

A: The originator of the PPR TLV MUST set the A bit in order to signal that the prefixes and PPR-IDs advertised in the PPR TLV are directly connected to the originators. If this bit is not set, this allows any other node in the network advertise this TLV on behalf of the originating node of the "OSPF Prefix". If PPR TLV is propagated to other areas the A-flag MUST be cleared. In case if the originating node of the prefix has to be disambiguated for any reason including, if it is a Multi Homed Prefix (MHP) or propagated to a different OSPF area, then PPR-Attribute Sub-TLV Source Router ID SHOULD be included.

Rsrvd - reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

3.1. OSPFv3 PPR-Prefix Sub-TLV

The structure of OSPFv3 PPR-Prefix, for which path description is attached to is as follows:

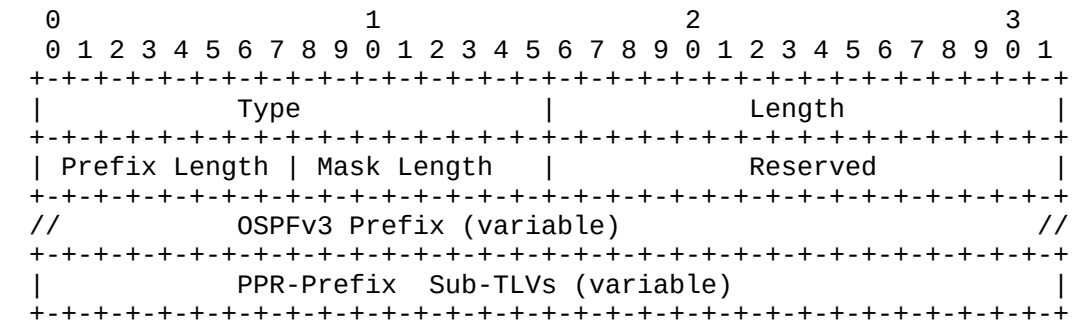


Figure 6: OSPFv3 PPR-Prefix Sub-TLV Format

- Type - TBD (See IANA for suggested value) from OSPFv3 PPR TLV Section 3 Sub-TLV registry.
- Length - Total length of the value field in bytes (variable).
- Prefix Len - contains the length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- Mask Length - The length of the prefix in bits. Only the most significant octets of the Prefix are encoded.
- OSPFv3 Prefix - represents the OSPFv3 prefix at the tail-end of the advertised PPR. For the address family IPv4 unicast, the prefix itself is encoded as a 32-bit value. The default route is represented by a prefix of length 0. For the address family (AF in OSPFv3 PPR TLV) in IPv6 unicast, the prefix, encoded as an even multiple of 32-bit words, padded with zeroed bits as necessary. This encoding consumes ((PrefixLength + 31) / 32) 32-bit words.
- PPR-Prefix Sub-TLVs - TBD. It has 2 octet type, 2 octet length and value field is defined per type.

3.2. OSPFv3 PPR-ID Sub-TLVs

This represents the actual data plane identifier in the packet and could be of any data plane as defined in PPR-ID-type field. Both OSPF Prefix and PPR-ID MUST belong to a same node in the network.

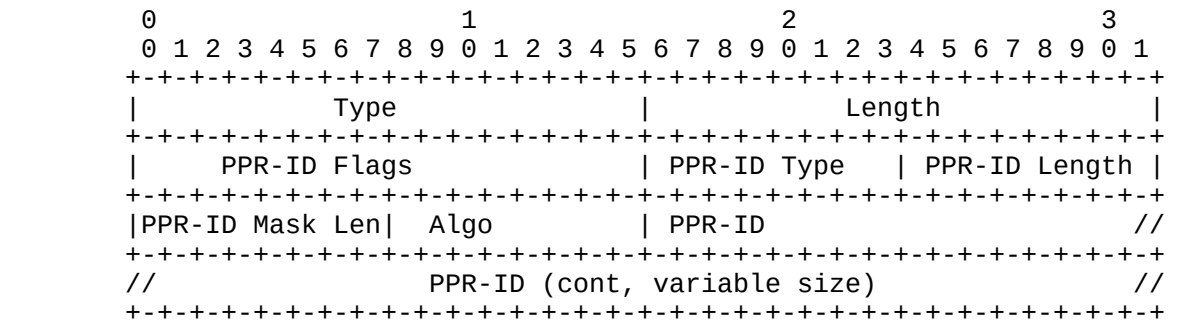


Figure 7: OSPFv3 PPR-ID Sub-TLV Format

- Type - TBD (See IANA for suggested value) from OSPFv3 PPR TLV Section 3 Sub-TLV registry.

- Length - Total length of the value field in bytes (variable).

- PPR-ID Type - Data plane type of PPR-ID. This is a new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 MPLS SID/Label

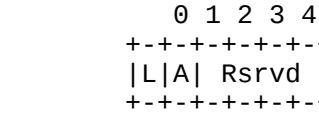
b. Type: 2 Native IPv4 Address/Prefix

c. Type: 3 Native IPv6 Address/Prefix

d. Type: 4 IPv6 SID in SRv6 with SRH

- PPR-ID Flags - 2 Octet field for PPR-ID flags:

PPR-ID Flags Format



1. L - If set, the PPR path is a Loose-PPR. If the flag is unset, then the path described is a Strict-PPR. A Strict-PPR lists every single node or adjacency in the path description from source to the destination.

2. A - If set, all non-PPR path nodes in the OSPF area MUST add a FIB entry for the PPR-ID with NH set to the shortest path NH for

the prefix being advertised. The use of this is TBD. By default this MUST be unset.

3. Reserved - Reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

- PPR-ID Length - Length of the PPR-ID field in octets and this depends on the PPR-ID type. See PPR-ID below for the length of this field and other considerations.

- PPR-ID Mask Len - It is applicable for only for PPR-ID Type 2, 3 and 4. For Type 1 this value MUST be set to zero. It contains the length of the PPR-ID Prefix in bits. Only the most significant octets of the Prefix are encoded. This is needed, if PPR-ID followed is an IPv4/IPv6 Prefix instead of 4/16 octet Address respectively.

- Algo - 1 octet value represents the SPF algorithm. Algorithm registry is as defined in [I-D.ietf-ospf-ospfv3-segment-routing-extensions].

- PPR-ID - This is the Preferred Path forwarding identifier that would be on the data packet. The value of this field is variable and it depends on the PPR-ID Type - for Type 1, this is and MPLS SID/Label. For Type 2 this is a 4 byte IPv4 address. For Type 3 this is a 16 byte IPv6 address. For Type 2 and Type 3 encoding is similar to OSPF Prefix as specified in Section 2.2. For Type 4, it is a 16 byte IPv6 SID.

3.3. OSPFv3 PPR-PDE Sub-TLV

This is a new Sub-TLV type in PPR TLV Section 3 and is called as PPR Path Description Element (PDE). PPR-PDEs are used to describe the path in the form of set of contiguous and ordered Sub-TLVs, where first Sub-TLV represents (the top of the stack in MPLS data plane or) first node/segment of the path. These set of ordered Sub-TLVs can have both topological SIDs and non-topological SIDs (e.g., service segments).

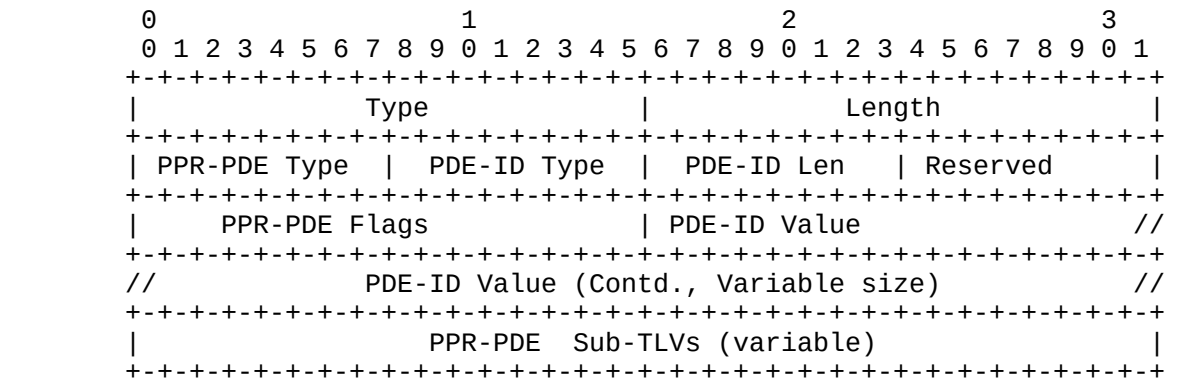


Figure 8: OSPFv3 PPR-PDE Sub-TLV Format

- o Type - TBD (See IANA for suggested value) from OSPFv3 PPR TLV Section 3 Sub-TLV registry.

- o Length - Total length of the value field in bytes (variable).

- o PPR-PDE Type - This is a new registry (TBD IANA) for this Sub-TLV and the defined types are as follows:

a. Type: 1 Topological

b. Type: 2 Non-Topological

- o PDE-ID Type - 1 Octet PDE-forwarding IDentifier Type. This is a new registry (TBD IANA) for this Sub-TLV and the defined types and corresponding PDE-ID Len, PDE-ID Value are as follows:

a. Type 1: SID/Label Sub-TLV as defined in [I-D.ietf-ospf-segment-routing-extensions]. PED-ID Len and PDE-ID Value fields are per Section 2.1 of the referenced document.

b. Type 2: SR-MPLS Prefix SID. PDE-ID Len and PDE-ID Value are same as Type 1.

c. Type 3: SR-MPLS Adjacency SID. PDE-ID Len and PDE-ID Value are same as Type 1.

d. Type 4: IPv4 Address. PDE-ID Len is 4 bytes and PDE-ID Value is 4 bytes IPv4 address encoded similar to IPv4 Prefix described in Section 2.2.

e. Type 5: IPv6 Address. PDE-ID Len is 16 bytes and PDE-ID Value is 16 bytes IPv6 address encoded similar to IPv6 Prefix described in Section 2.2.

f. Type 6: SRv6 Node SID as defined in [I-D.li-ospf-ospfv3-srv6-extensions]. PDE-ID Len and PDE-ID Value are as defined in SRv6 SID.

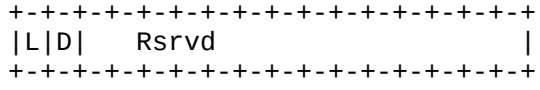
g. Type 7: SRv6 Adjacency-SID. PDE-ID Len and PDE-ID Value are as defined in Type 6.

o PDE-ID Len - 1 Octet. Length of PDE-ID field.

o Reserved - 1 Octet reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

o PPR-PDE Flags - 2 Octet flags for this TLV are described below:

PPR-PDE Flags Format



1. L - This bit indicates the type of next "Topological PDE-ID" in the path description and overrides the L bit in Section 3.2. If set, the next PDE is Loose. If this flag is unset, the next Topological PDE is Strict Type.

2. D - By default this bit MUST be unset. This bit MUST be set only for PPR-PDE Type is Topological and this PDE represents the PDE-ID corresponding to the PPR-Prefix Section 3.1.

3. Rsrvd - Reserved bits for future use. Reserved bits MUST be reset on transmission and ignored on receive.

o PPR-PDE Sub-TLVs - TBD. It has 2 octet type, 2 octet length and value field is defined per type.

3.4. OSPFv3 PPR-Attributes Sub-TLV

PPR-Attribute Sub-TLVs describe the attributes of the path. The following Sub-TLVs draw from a new registry for Sub-TLV numbers; this registry is to be created by IANA, and administered using the first come first serve process:

- Type 1 (Suggested Value - IANA TBD): This is Packet Traffic accounting Sub-TLV. Length 0 No value field. Specifies to create a counter to count number of packets forwarded on this PPR-ID on each node in the path description.

- Type 2 (Suggested Value - IANA TBD): This is Traffic statistics in Bytes Sub-TLV. Length 0 No value field. Specifies to create a counter to count number of bytes forwarded on this PPR-ID specified in the network header (e.g. IPv4, IPv6) on each node in the path description.

- Type 3 (Suggested Value - IANA TBD): PPR-Metric Sub-TLV. Length 4 bytes, and Value is metric of this path represented through the PPR-ID. Different nodes can advertise the same PPR-ID for the same Prefix with a different set of PPR-PDE Sub-TLVs and the receiving node MUST consider the lowest metric value (TBD more, on what happens when metric is same for two different set of PPR-PDE Sub-TLVs).

4. Other Considerations

Please refer to [I-D.chunduri-isis-preferred-path-routing] section 4, 5, 6 and 7.

5. Acknowledgements

Thanks to Richard Li, Alex Clemm, Padma Pillay-Esnault, Toerless Eckert, Kiran Makhijani and Lin Han for initial discussions on this topic. Thanks to Kevin Smith and Stephen Johnson for various deployment scenarios applicability from ETSI WGs perspective. Authors also acknowledge Alexander Vainshtein for detailed discussions and suggestions on this topic.

Earlier versions of draft-ietf-ospf-segment-routing-extensions have a mechanism to advertise EROs through Binding SID.

6. IANA Considerations

This document requests the following new TLV in IANA OSPFv2 and OSPFv3 TLV code-point registry as specified in Section 2 Section 3 respectively .

TLV #	Name
TBD	PPR TLV

This document also requests IANA to create new registries for PPR TLV Flags field, PPR Flags, and PPR Sub-TLVs in PPR TLV as described in Section 2 and Section 3.

7. Security Considerations

Existing security extensions as described in [RFC2328] and [RFC7684] apply to the extensions specified in this document. While OSPF is under a single administrative domain, there can be deployments where potential attackers have access to one or more networks in the OSPF routing domain. In these deployments, stronger authentication mechanisms such as those specified in [RFC7474] SHOULD be used.

Advertisement of the additional information defined in this document introduces no new security concerns in OSPF protocol. However as this extension is related to SR-MPLS and SRH data planes as defined in [I-D.ietf-spring-segment-routing], those particular data plane security considerations does apply here.

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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[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>.

8.2. Informative References

[I-D.chunduri-lsr-isis-preferred-path-routing] Chunduri, U., Li, R., White, R., Tantsura, J., Contreras, L., and Y. Qu, "Preferred Path Routing (PPR) in IS-IS", draft-chunduri-lsr-isis-preferred-path-routing-00 (work in progress), June 2018.

[I-D.ietf-6man-segment-routing-header]
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-14 (work in progress), June 2018.

[I-D.ietf-ospf-ospfv3-lsa-extend]
Lindem, A., Roy, A., Goethals, D., Vallem, V., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-23 (work in progress), January 2018.

[I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Filsfils, C., Previdi, S., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-13 (work in progress), May 2018.

[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-25 (work in progress), April 2018.

[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.li-ospf-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", draft-li-ospf-ospfv3-srv6-extensions-01 (work in progress), March 2018.

[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>.

[RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <https://www.rfc-editor.org/info/rfc7474>.

[RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", RFC 7684, DOI 10.17487/RFC7684, November 2015, <https://www.rfc-editor.org/info/rfc7684>.

Authors' Addresses

Uma Chunduri
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: uma.chunduri@huawei.com

Yingzhen Qu
Huawei USA
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yingzhen.qu@huawei.com

Russ White
LinkedIn
Oak Island, NC 28465
USA

Email: russ@riw.us

Jeff Tantsura
Nuage Networks
755 Ravendale Drive
Mountain View, CA 94043
USA

Email: jefftant.ietf@gmail.com

Luis M. Contreras
Telefonica
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2018

J. Dong
S. Bryant
Huawei Technologies

June 20, 2018

IGP Extensions for Segment Routing based Enhanced VPN
draft-dong-lsr-sr-enhanced-vpn-00

Abstract

Enhanced VPN (VPN+) is an enhancement to VPN services to support the needs of new applications, particularly including the applications that are associated with 5G services. These applications require better isolation and have more stringent performance requirements than that can be provided with traditional overlay VPNs. An enhanced VPN may form the underpin of 5G transport network slicing, and will also be of use in its own right. This document describes how Multi-Topology Routing (MTR) as described in RFC 5120 and RFC 4915, can be extended to signal the resources allocated in the underlay network to construct the virtual networks for enhanced VPN services, together with the Segment Routing Identifiers (SIDs) used to identify and access the network resources allocated for the virtual networks in the data plane.

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 22, 2018.

Table of Contents

1. Introduction

The framework for an enhanced virtual private network (VPN+) is described in [I-D.bryant-rtgwg-enhanced-vpn].

Driven largely by needs arising from the 5G mobile network design, the concept of network slicing has gained traction. There is a need to create a VPN service with enhanced isolation and performance characteristics. Specifically, there is a need for a transport network to support a set of virtual networks, each of which provides the client with some dedicated (private) network resources drawn from

a shared pool. The tenant of such a virtual network can require a degree of isolation and performance that previously could only be satisfied by dedicated networks. Additionally the tenant may ask for some level of control of their virtual networks e.g. to customize the service paths in their network slices.

These properties cannot be met with pure overlay networks, as they require tighter coordination and integration between the underlay and the overlay network. [I-D.bryant-rtgwg-enhanced-vpn] provides the framework of enhanced VPN and describes the candidate component technologies. [I-D.dong-spring-sr-for-enhanced-vpn] describes how segment routing (SR) [I-D.ietf-spring-segment-routing] is used to construct the required virtual networks with the network resources allocated for enhanced VPN services.

This document describes how Multi-Topology Routing (MTR), as described in [RFC5120] [RFC4915] , is extended to signal the resources allocated in the underlay to construct the virtual networks for enhanced VPN services, together with the segment routing identifiers used to identify and access the resource allocated for different virtual networks in the data plane.

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 RFC 2119 [RFC2119].

3. Overview of Approach

To meet the requirement of enhance VPN services, a number of virtual networks can be created, each representing a subset of the underlay network topology and resources to be used by a specific customer. In a 5G context, each virtual network is considered as a network slice which serves one slice tenant. Depending on the service requirements, differernt virtual networks can either share the same physical links or nodes, or use separate links or nodes in the network, while the required level of isolation and performance SHOULD be guaranteed in both cases.

IGP multi-topology routing can be seen as a candidate mechanism to create multiple network topologies in one network. Different from the traditional multi-topology mechanism which only provides logical topological isolation, in the proposed mechanism network resources can be partitioned and allocated to different virtual network topologies to meet the isolation and performance requirements of enhanced VPN. Service in one virtual topology can be instructed to be processed using the network resources allocated to this virtual

topology. This is achieved by using multi-topology together with segment routing, and extending the SR paradigm to use Segment Identfiers (SIDs) to identify different set of resources allocated from a particular network element (e.g. link or node). Different set of SIDs are associated with different virtual topologies, and are used to create the SID lists in different topologies. In some cases it is also possible for several virtual network topologies to share some network resources, this can be achieved by using the same SR SIDs between those topologies. The detailed mechanism of resource sharing will be described in a future version.

Within one SR virtual network, one or more type of services can be deployed using the resources allocated to that topology, some of which may have different characteristics and require dedicated resources or special treatment. This is similar to the DS-TE model of the RSVP-TE based mechanism. The concept is similar to the DS-TE model [RFC4124] of RSVP-TE based mechanism, while in this case the SR paradigm is applied, which avoids the introduction of per-path state into the network.

In general this approach applies to both IS-IS and OSPF, while the specific protocol extensions and encodings are different. In the current version of this document, the required IS-IS extensions are described. The required OSPF extensions will be described in a future version.

4. SR Virtual Topology with Resource Guarantee

As described in [I-D.ietf-isis-segment-routing-extensions], the IS-IS TLV-222 (MT-ISN) and TLV-223 (MT IS Neighbor Attribute) have been enhanced to carry the Adj-SID sub-TLV, and TLV-235 (Multitopology IPv4 Reachability) and TLV-237 (Multitopology IPv6 IP Reachability) have been enhanced to carry the Prefix-SID sub-TLV. With these enhancements, dedicated Segment Identifiers (SIDs) can be assigned for each SR virtual network topology.

This section specifies the necessary extensions to enable the deployment of resource guaranteed SR virtual topologies. Each virtual topology can be allocated with a particular partition of network resources from the underlay network, the SIDs associated with each SR virtual topology are used to identify the set of resources allocated from the network elements.

4.1. Topology specific Link Resource Allocation and Identification

A network link can participate in one or multiple SR virtual topologies, each virtual topology is assigned with a dedicated adj-SID. In order to describe the amount of link resource allocated to a

particular SR virtual topology, a new IS-IS sub-TLV called "SR Bandwidth" sub-TLV is defined:

The SR-Bandwidth sub-TLV is an optional sub-TLV carrying the aggregated bandwidth allocated to a particular SR adj-SID, which is associated witha particular virtual topology. In the data plane, the allocated bandwidth and the associated functional components are identified by the adj-SID of the virtual topology. This sub-TLV may be advertised as a sub-TLV of the following TLVs:

TLV-22 (Extended IS reachability) [RFC5305]

TLV-23 (IS Neighbor Attribute) [RFC5311]

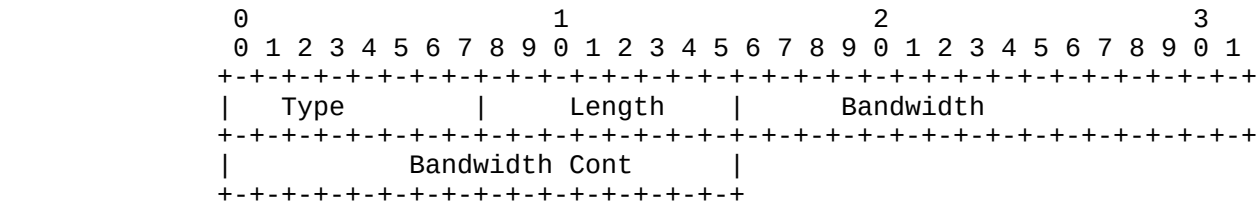
TLV-141 (inter-AS reachability information) [RFC5316]

TLV-222 (Multitopology IS)[RFC5120]

TLV-223 (Multitopology IS Neighbor Attribute) [RFC5311]

The SR bandwidth sub-TLV can appear at most once for a particular topology. Multiple SR Bandwidth sub-TLVs MAY be associated with a single IS neighbor.

The following format is defined for the SR Bandwidth sub-TLV:



SR Bandwidth sub-TLV

where:

Type: TBD, to be assigned by IANA.

Length: variable.

The SR bandwidth is encoded in 32 bits in IEEE floating point format. The units are bytes (not bits!) per second.

[I-D.ietf-teas-sr-rsvp-coexistence-rec] describes several options for traffic engineering in networks where RSVP-TE and SR LSPs coexist. Note that section 3.1 of [I-D.ietf-teas-sr-rsvp-coexistence-rec] proposes to partition the network bandwidth between RSVP-TE and SR.

The can be considered as a special case of creating one default SR virtual topology with dedicated bandwidth allocated, so that the network resources and operation of SR are isolated from the RSVP-TE based LSPs.

4.2. Topology specific Node Resource Allocation and Identification

A network node can participate in one or multiple SR virtual topologies, each virtual topology is assigned with a dedicated node-SID. In SR loose path forwarding, the topology specific node-SIDs can be used by transit network nodes to identify the virtual topology the packet belongs to, so as to steer the packet through the set of link resources allocated for the identified virtual topology. A prefix-SID sub-TLV describing the dedicated node-SID for each virtual topology is needed, this is supported in [I-D.ietf-isis-segment-routing-extensions].

In addition, similar to the allocation of link resource to virtual topologies, it is possible to allocate a subset of nodal resources for a particular virtual topology to ensure end-to-end service delivery. The nodal resources can be identified by the topology specific node-SIDs, so that in data plane the node SIDs can be used to steer a packet through the set of nodal resources allocated to this topology. Optional sub-TLVs describing the allocated resources at the node level for a particular virtual topology may be defined in future. The specification of nodal resources is for further study.

5. Multiple Services in SR Virtual Topology

Within one SR virtual topology, one or more types of service can be deployed using the resources allocated to this virtual topology. Each service type can have specific resource constraints and characteristics. The concept is similar to the DS-TE model [RFC4124] of RSVP-TE based mechanism, while in this case the SR paradigm is applied, which avoids the introduction of per-flow state into the network.

Some mechanism is needed to identify different service types and specify the different service characteristics within one virtual topology. The detailed protocol extensions will be provided in a future version.

5.1. Common Service Types

A service type is fundamentally the sum of the properties of a group of services. The authors considered specifically creating a number of specific service types within the protocol but concluded that this

was meaningless. The following sections show how a number of well known service types can be constructed.

5.1.1. Best Effort

Best effort service can be the only service type in a particular virtual topology. In this case, all of the resources allocated to this virtual topology instance are available to the best effort services.

Where there are multiple service types being carried in a virtual topology, best effort service will be transmitted over the links and nodes when there is an opportunity. The maximum resources which can be used by best effort service may be constrained to a subset of the topology resource. The Traffic Class (TC) of the best effort service SHOULD be set to lower than any other service types.

In the data plane, the SID and the Traffic Class value in the packet can be used to identify the service type and steer the best effor packets into the correct forwarding resources, such as queues.

Best effort services may or may not be protected at the discretion of the network operator.

5.1.2. Assured Bandwidth

An Assured Bandwidth service is one in which the bandwidth is assured but the latency is not. Thus, some bandwidth can be allocated to the assured bandwidth service, and traffic up to that bandwidth will be transmitted over the service, but the traffic may be delayed by other traffic.

It is likely that the assured bandwidth service will be carried in a virtual topology together with other service types, such as the best effort service. The maximum resources which can be used by assured bandwidth service SHOULD be constrained to a subset of the topology resource.

There will frequently be more than one assured bandwidth service running on a topology, and the Traffic Class (TC) could be used to determine how the various services compete for access to the link. Whilst the bandwidth is assured over the long term, over the short term it is not and such services will interact with similar and lower service classes in such a way that packet delay and jitter is not assured.

In the data plane, the SID and the Traffic Class value in the packet can be used to identify the service type and priority and steer the

assured bandwidth service packets into the correct forwarding resources, such as queues.

5.1.3. Deterministic

A Deterministic service is a service that may have controlled delay/jitter characteristics and/or an enhanced packet delivery assurance. Delay/Jitter may be addressable through the provision of sufficient bandwidth or it may require some form of packet scheduling. Enhanced delivery assurance may require the use of packet replication and elimination mechanism. The design of a deterministic network is discussed in [I-D.ietf-detnet-architecture]. Note that delay protection and delivery protection are orthogonal characteristics and a service may provide just one of the characteristics or it may provide both.

The details of a deterministic service will be provided in a future version. Such a service may be specified using the TLVs defined in [I-D.geng-detnet-info-distribution]

6. Topology and Algorithm

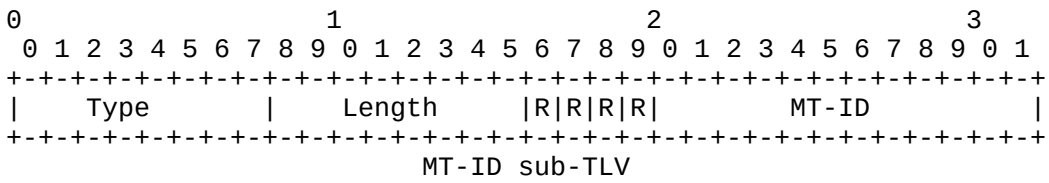
In the proposed mechanism, SR is used with IGP multi-topology to create one or more SR virtual topologies, each associated with a set of network resources allocated for the virtual topology. The service paths used between nodes in one virtual topology are not constrained to be shorted path by IGP metric, and can be any non-looping path that best suits the needs of the service. These paths may be imposed by the network controller, or calculated using a distributed method. For example, different SR algorithms as defined in [I-D.ietf-isis-segment-routing-extensions] can be used within one virtual topology. The flex-algo mechanism defined in [I-D.ietf-lsr-flex-algo] may also be used in one virtual topology to meet different service requirements.

7. SRv6 Considerations

The mechanisms to create virtual network topologies with allocated resources using an SRv6 data-plane are similar to SR with an MPLS data plane, although there are some differences to be considered. This section specifies the necessary protocol extensions to enable SRv6 with multi-topology and the mechanisms defined in this document. Detailed method of operating enhanced VPN over an SRv6 data-plane will be described in a future version.

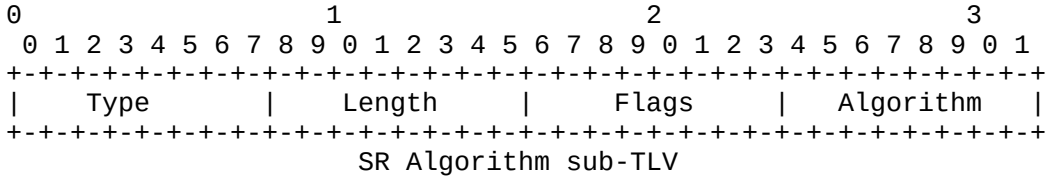
In [I-D.bashandy-isis-srv6-extensions], the SRv6 node SID TLV is defined as a top-level TLV, which cannot be carried under the MT IPv6 IP Reach TLV (type 237). In order to specify the association between

SRv6 node SIDs and differernt virtual topologies, a new sub-TLV called "MT-ID sub-TLV" under the SRv6 node SID TLV is introduced. The format of the sub-TLV is as follows:



MT-ID sub-TLV

In addition, in order to advertise multiple algorithms used for calculating reachability to nodes within a particular topology, a new sub-TLV called "SR algorithm sub-TLV" under the SRv6 node SID TLV is introduced. The format of the sub-TLV is as follows:



SR Algorithm sub-TLV

In order to advertise the SRv6 adj-SIDs associated with different topologies the network node participates in, the SRv6 Adjacency SID sub-TLV as defined in [I-D.bashandy-isis-srv6-extensions] MUST be carried in the MT Intermediate Systems TLV (type 222). The SR bandwidth sub-TLV as defined in this document SHOULD also be carried in the MT Intermediate Systems TLV.

8. Fast Repair

In some instances it is desirable to provide some form of fast repair for a failed link or node. The methods available fall into two categories, end-to-end, for example 1+1, and IP fast reroute. Which ever of these is used it is desirable that the repair path provides the same level of service to the tenant as the tenant’s normal service. This would mean that the repair path needs to be constrained to the tenant’s topology, or to some repair topology reserved exclusively for that tenant for the duration of the repair. The normal way that IPFRR operates is that the point of local repair (PLR) calculates the repair path based on the information flooded by the routing protocol. How the PLR can maintain the level of service through the repair is for further study.

9. LAN interface

The use of multi-point to multi-point (MP2MP) interfaces is currently out of scope for this design.

A LAN interface MUST be used in point to point mode.

Note support for MP2MP may be needed in the future, and this is for further study.

10. Security Considerations

This document introduces no additional security vulnerabilities to IS-IS and OSPF.

The mechanism proposed in this document is subject to the same vulnerabilities as any other protocol that relies on IGPs.

11. IANA Considerations

This document requests IANA to allocate a sub-TLV type as defined in Section 4 from "Sub-TLVs for TLVs 22, 23, 25, 141, 222 and 223" registry.

Value	Description	Reference
TBA1	SR bandwidth sub-TLV	This document

Per TLV information where SR bandwidth sub-TLV can be part of:

TLV	22	23	25	141	222	223
	y	y	n	y	y	y

This document requests IANA to allocate 2 sub-TLVs type as defined in Section 7 from the "Sub-TLVs for TLVs 27, 135, 235, 236 and 237" registry.

Value	Description	Reference
TBA2	MT-ID sub-TLV	This document
TBA3	SR Algorithm sub-TLV	This document

Per TLV information where the sub-TLVs can be part of:

TLV	27	135	235	236	237
TBA2	y	n	n	n	n
TBA3	y	n	n	n	n

12. Acknowledgments

The authors would like to thank Mach Chen and Robin Li for the review and discussion of this document.

13. References

13.1. Normative References

[I-D.dong-spring-sr-for-enhanced-vpn] Dong, J., Bryant, S., Li, Z., and T. Miyasaka, "Segment Routing for Enhanced VPN Service", draft-dong-spring-sr-for-enhanced-vpn-00 (work in progress), March 2018.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[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>.

[RFC5311] McPherson, D., Ed., Ginsberg, L., Previdi, S., and M. Shand, "Simplified Extension of Link State PDU (LSP) Space for IS-IS", RFC 5311, DOI 10.17487/RFC5311, February 2009, <https://www.rfc-editor.org/info/rfc5311>.

[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>.

13.2. Informative References

[I-D.bashandy-isis-srv6-extensions]
Ginsberg, L., Bashandy, A., Filsfils, C., and B. Decraene, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-01 (work in progress), September 2017.

[I-D.bryant-rtgwg-enhanced-vpn]
Bryant, S. and J. Dong, "Enhanced Virtual Private Networks (VPN+)", draft-bryant-rtgwg-enhanced-vpn-01 (work in progress), October 2017.

[I-D.geng-detnet-info-distribution]
Geng, X. and M. Chen, "IGP-TE Extensions for DetNet Information Distribution", draft-geng-detnet-info-distribution-01 (work in progress), September 2017.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-04 (work in progress), October 2017.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-15 (work in progress), December 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-00 (work in progress), May 2018.

[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.ietf-teas-sr-rsvp-coexistence-rec]
Sitaraman, H., Beeram, V., Minei, I., and S. Sivabalan, "Recommendations for RSVP-TE and Segment Routing LSP co-existence", draft-ietf-teas-sr-rsvp-coexistence-rec-04 (work in progress), May 2018.

[RFC4124] Le Faucheur, F., Ed., "Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering", RFC 4124, DOI 10.17487/RFC4124, June 2005, <https://www.rfc-editor.org/info/rfc4124>.

Authors' Addresses

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Stewart Bryant
Huawei Technologies

Email: stewart.bryant@gmail.com

IS-IS for IP Internets
Internet-Draft
Obsoletes: 5306 (if approved)
Intended status: Standards Track
Expires: December 30, 2018

L. Ginsberg
P. Wells
Cisco Systems, Inc.

June 28, 2018

Restart Signaling for IS-IS
draft-ginsberg-isis-rfc5306bis-01

Abstract

This document describes a mechanism for a restarting router to signal to its neighbors that it is restarting, allowing them to reestablish their adjacencies without cycling through the down state, while still correctly initiating database synchronization.

This document additionally describes a mechansim for a router to signal its neighbors that it is preparing to initiate a restart while maintaining forwarding plane state. This allows the neighbors to maintain their adjacencies until the router has restarted, but also allows the neighbors to bring the adjacencies down in the event of other topology changes.

This document additionally describes a mechanism for a restarting router to determine when it has achieved Link State Protocol Data Unit (LSP) database synchronization with its neighbors and a mechanism to optimize LSP database synchronization, while minimizing transient routing disruption when a router starts.

This document obsoletes RFC 5306.

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 December 30, 2018.

Table of Contents

1. Overview

The Intermediate System to Intermediate System (IS-IS) routing protocol [RFC1195] [ISO10589] is a link state intra-domain routing protocol. Normally, when an IS-IS router is restarted, temporary disruption of routing occurs due to events in both the restarting router and the neighbors of the restarting router.

The router that has been restarted computes its own routes before achieving database synchronization with its neighbors. The results of this computation are likely to be non-convergent with the routes computed by other routers in the area/domain.

Neighbors of the restarting router detect the restart event and cycle their adjacencies with the restarting router through the down state. The cycling of the adjacency state causes the neighbors to regenerate their LSPs describing the adjacency concerned. This in turn causes a temporary disruption of routes passing through the restarting router.

In certain scenarios, the temporary disruption of the routes is highly undesirable. This document describes mechanisms to avoid or minimize the disruption due to both of these causes.

When an adjacency is reinitialized as a result of a neighbor restarting, a router does three things:

1. It causes its own LSP(s) to be regenerated, thus triggering SPF runs throughout the area (or in the case of Level 2, throughout the domain).

2. It sets SRMflags on its own LSP database on the adjacency concerned.

3. In the case of a Point-to-Point link, it transmits a complete set of Complete Sequence Number PDUs (CSNPs), over the adjacency.

In the case of a restarting router process, the first of these is highly undesirable, but the second is essential in order to ensure synchronization of the LSP database.

The third action above minimizes the number of LSPs that must be exchanged and, if made reliable, provides a means of determining when the LSP databases of the neighboring routers have been synchronized. This is desirable whether or not the router is being restarted (so

that the overload bit can be cleared in the router's own LSP, for example).

This document describes a mechanism for a restarting router to signal that it is restarting to its neighbors, and allow them to reestablish their adjacencies without cycling through the down state, while still correctly initiating database synchronization.

This document additionally describes a mechanism for a restarting router to determine when it has achieved LSP database synchronization with its neighbors and a mechanism to optimize LSP database synchronization and minimize transient routing disruption when a router starts.

It is assumed that the three-way handshake [RFC5303] is being used on Point-to-Point circuits.

2. Approach

2.1. Timers

Three additional timers, T1, T2, and T3, are required to support the functionality defined in this document.

An instance of the timer T1 is maintained per interface, and indicates the time after which an unacknowledged (re)start attempt will be repeated. A typical value might be 3 seconds.

An instance of the timer T2 is maintained for each LSP database (LSPDB) present in the system, i.e., for a Level 1/2 system, there will be an instance of the timer T2 for Level 1 and an instance for Level 2. This is the maximum time that the system will wait for LSPDB synchronization. A typical value might be 60 seconds.

A single instance of the timer T3 is maintained for the entire system. It indicates the time after which the router will declare that it has failed to achieve database synchronization (by setting the overload bit in its own LSP). This is initialized to 65535 seconds, but is set to the minimum of the remaining times of received IS-IS Hellos (IIHs) containing a restart TLV with the Restart Acknowledgement (RA) set and an indication that the neighbor has an adjacency in the "UP" state to the restarting router.

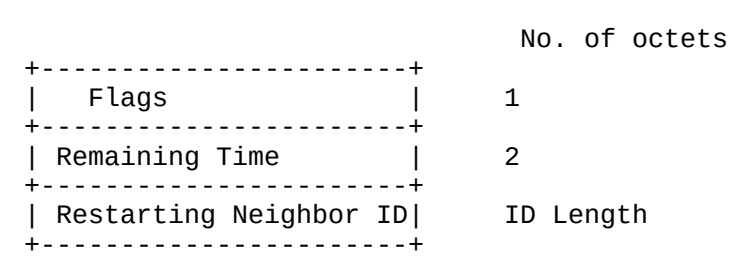
NOTE: The timer T3 is only used by a restarting router.

2.2. Restart TLV

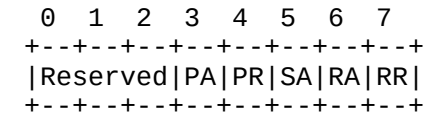
A new TLV is defined to be included in IIH PDUs. The presence of this TLV indicates that the sender supports the functionality defined in this document and it carries flags that are used to convey information during a (re)start. All IIHs transmitted by a router that supports this capability MUST include this TLV.

Type 211

Length: Number of octets in the Value field (1 to (3 + ID Length))
Value



Flags (1 octet)



RR - Restart Request
RA - Restart Acknowledgement
SA - Suppress adjacency advertisement
PR - Restart is planned
PA - Planned restart acknowledgement

(Note: Remaining fields are required when the RA bit is set.)
Remaining Time (2 octets)

Remaining holding time (in seconds)

Restarting Neighbor System ID (ID Length octets)

The System ID of the neighbor to which an RA refers. Note: Implementations based on earlier versions of this document may not include this field in the TLV when the RA is set. In this case, a router that is expecting an RA on a LAN circuit SHOULD assume that the acknowledgement is directed at the local system.

2.2.1. Use of RR and RA Bits

The RR bit is used by a (re)starting router to signal to its neighbors that a (re)start is in progress, that an existing adjacency SHOULD be maintained even under circumstances when the normal operation of the adjacency state machine would require the adjacency to be reinitialized, to request a set of CSNPs, and to request setting of the SRMflags.

The RA bit is sent by the neighbor of a (re)starting router to acknowledge the receipt of a restart TLV with the RR bit set.

When the neighbor of a (re)starting router receives an IIH with the restart TLV having the RR bit set, if there exists on this interface an adjacency in state "UP" with the same System ID, and in the case of a LAN circuit, with the same source LAN address, then, irrespective of the other contents of the "Intermediate System Neighbors" option (LAN circuits) or the "Point-to-Point Three-Way Adjacency" option (Point-to-Point circuits):

a. the state of the adjacency is not changed. If this is the first IIH with the RR bit set that this system has received associated with this adjacency, then the adjacency is marked as being in "Restart mode" and the adjacency holding time is refreshed -- otherwise, the holding time is not refreshed. The "remaining time" transmitted according to (b) below MUST reflect the actual time after which the adjacency will now expire. Receipt of a normal IIH with the RR bit reset will clear the "Restart mode" state. This procedure allows the restarting router to cause the neighbor to maintain the adjacency long enough for restart to successfully complete, while also preventing repetitive restarts from maintaining an adjacency indefinitely. Whether or not an adjacency is marked as being in "Restart mode" has no effect on adjacency state transitions.

b. immediately (i.e., without waiting for any currently running timer interval to expire, but with a small random delay of a few tens of milliseconds on LANs to avoid "storms") transmit over the corresponding interface an IIH including the restart TLV with the RR bit clear and the RA bit set, in the case of Point-to-Point adjacencies having updated the "Point-to-Point Three-Way Adjacency" option to reflect any new values received from the (re)starting router. (This allows a restarting router to quickly acquire the correct information to place in its hellos.) The "Remaining Time" MUST be set to the current time (in seconds) before the holding timer on this adjacency is due to expire. If the corresponding interface is a LAN interface, then the Restarting Neighbor System ID SHOULD be set to the System ID of

the router from which the IIH with the RR bit set was received. This is required to correctly associate the acknowledgement and holding time in the case where multiple systems on a LAN restart at approximately the same time. This IIH SHOULD be transmitted before any LSPs or SNPs are transmitted as a result of the receipt of the original IIH.

c. if the corresponding interface is a Point-to-Point interface, or if the receiving router has the highest LnRouterPriority (with the highest source MAC (Media Access Control) address breaking ties) among those routers to which the receiving router has an adjacency in state "UP" on this interface whose IIHs contain the restart TLV, excluding adjacencies to all routers which are considered in "Restart mode" (note the actual DIS is NOT changed by this process), initiate the transmission over the corresponding interface of a complete set of CSNPs, and set SRMflags on the corresponding interface for all LSPs in the local LSP database.

Otherwise (i.e., if there was no adjacency in the "UP" state to the System ID in question), process the IIH as normal by reinitializing the adjacency and setting the RA bit in the returned IIH.

2.2.2. Use of the SA Bit

The SA bit is used by a starting router to request that its neighbor suppress advertisement of the adjacency to the starting router in the neighbor’s LSPs.

A router that is starting has no maintained forwarding function state. This may or may not be the first time the router has started. If this is not the first time the router has started, copies of LSPs generated by this router in its previous incarnation may exist in the LSP databases of other routers in the network. These copies are likely to appear "newer" than LSPs initially generated by the starting router due to the reinitialization of LSP fragment sequence numbers by the starting router. This may cause temporary blackholes to occur until the normal operation of the update process causes the starting router to regenerate and flood copies of its own LSPs with higher sequence numbers. The temporary blackholes can be avoided if the starting router’s neighbors suppress advertising an adjacency to the starting router until the starting router has been able to propagate newer versions of LSPs generated by previous incarnations.

When a router receives an IIH with the restart TLV having the SA bit set, if there exists on this interface an adjacency in state "UP" with the same System ID, and in the case of a LAN circuit, with the same source LAN address, then the router MUST suppress advertisement

of the adjacency to the neighbor in its own LSPs. Until an IIH with the SA bit clear has been received, the neighbor advertisement MUST continue to be suppressed. If the adjacency transitions to the "UP" state, the new adjacency MUST NOT be advertised until an IIH with the SA bit clear has been received.

Note that a router that suppresses advertisement of an adjacency MUST NOT use this adjacency when performing its SPF calculation. In particular, if an implementation follows the example guidelines presented in [ISO10589], Annex C.2.5, Step 0:b) "pre-load TENT with the local adjacency database", the suppressed adjacency MUST NOT be loaded into TENT.

2.2.3. Use of PR and PA Bits

The PR bit is used by a router which is planning to initiate a restart to signal to its neighbors that it will be restarting.

The PA bit is sent by the neighbor of a router planning to restart to acknowledge receipt of a restart TLV with the PR bit set.

When the neighbor of a router planning a restart receives an IIH with the restart TLV having the PR bit set, if there exists on this interface an adjacency in state "UP" with the same System ID, and in the case of a LAN circuit, with the same source LAN address, then:

a. if this is the first IIH with the PR bit set that this system has received associated with this adjacency, then the adjacency is marked as being in "Planned Restart state" and the adjacency holding time is refreshed -- otherwise, the holding time is not refreshed. The "remaining time" transmitted according to (b) below MUST reflect the actual time after which the adjacency will now expire. Receipt of a normal IIH with the PR bit reset will clear the "Planned Restart mode" state. This procedure allows the router planning a restart to cause the neighbor to maintain the adjacency long enough for restart to successfully complete. Whether or not an adjacency is marked as being in "Planned Restart mode" has no effect on adjacency state transitions.

b. immediately (i.e., without waiting for any currently running timer interval to expire, but with a small random delay of a few tens of milliseconds on LANs to avoid "storms") transmit over the corresponding interface an IIH including the restart TLV with the PR bit clear and the PA bit set. The "Remaining Time" MUST be set to the current time (in seconds) before the holding timer on this adjacency is due to expire. If the corresponding interface is a LAN interface, then the Restarting Neighbor System ID SHOULD be set to the System ID of the router from which the IIH with the

PR bit set was received. This is required to correctly associate the acknowledgement and holding time in the case where multiple systems on a LAN are planning a restart at approximately the same time.

While a control plane restart is in progress it is expected that the restarting router will be unable to respond to topology changes. It is therefore useful to signal a planned restart (if the forwarding plane on the restarting router is maintained) so that the neighbors of the restarting router can determine whether it is safe to maintain the adjacency if other topology changes occur prior to the completion of the restart. Signalling a planned restart in the absence of maintained forwarding plane state is likely to lead to significant traffic loss and MUST NOT be done.

Neighbors of the router which has signaled planned restart SHOULD maintain the adjacency in a planned restart state until it receives an IIH with the RR bit set, receives an IIH with both PR and RR bits clear, or the adjacency holding time expires - whichever occurs first.

While the adjacency is in planned restart state the following actions MAY be taken:

a. If additional topology changes occur, the adjacency which is in planned restart state MAY be brought down even though the hold time has not yet expired. Given that the neighbor which has signaled a planned restart is not expected to update its forwarding plane in response to signaling of the topology changes (since it is restarting) traffic which transits that node is at risk of being improperly forwarded. On a LAN circuit, if the router in planned restart state is the DIS at any supported level, the adjacency(ies) SHOULD be brought down whenever any LSP update is either generated or received so as to trigger a new DIS election. Failure to do so will compromise the reliability of the Update Process on that circuit. What other criteria are used to determine what topology changes will trigger bringing the adjacency down is a local implementation decision.

b. If a BFD session to the neighbor which signals a planned restart is in the UP state and subsequently goes DOWN, the event MAY be ignored since it is possible this is an expected side effect of the restart. Use of the Control Plane Independent state as signalled in BFD control packets [RFC5880] SHOULD be considered in the decision to ignore a BFD Session DOWN event

c. On a Point-to-Point circuit, transmission of LSPs, CSNPs, and PSNPs MAY be suppressed. It is expected that the PDUs will not be received.

2.3. Adjacency (Re)Acquisition

Adjacency (re)acquisition is the first step in (re)initialization. Restarting and starting routers will make use of the RR bit in the restart TLV, though each will use it at different stages of the (re)start procedure.

2.3.1. Adjacency Reacquisition during Restart

The restarting router explicitly notifies its neighbor that the adjacency is being reacquired, and hence that it SHOULD NOT reinitialize the adjacency. This is achieved by setting the RR bit in the restart TLV. When the neighbor of a restarting router receives an IIH with the restart TLV having the RR bit set, if there exists on this interface an adjacency in state "UP" with the same System ID, and in the case of a LAN circuit, with the same source LAN address, then the procedures described in Section 3.2.1 are followed.

A router that does not support the restart capability will ignore the restart TLV and reinitialize the adjacency as normal, returning an IIH without the restart TLV.

On restarting, a router initializes the timer T3, starts the timer T2 for each LSPDB, and for each interface (and in the case of a LAN circuit, for each level) starts the timer T1 and transmits an IIH containing the restart TLV with the RR bit set.

On a Point-to-Point circuit, the restarting router SHOULD set the "Adjacency Three-Way State" to "Init", because the receipt of the acknowledging IIH (with RA set) MUST cause the adjacency to enter the "UP" state immediately.

On a LAN circuit, the LAN-ID assigned to the circuit SHOULD be the same as that used prior to the restart. In particular, for any circuits for which the restarting router was previously DIS, the use of a different LAN-ID would necessitate the generation of a new set of pseudonode LSPs, and corresponding changes in all the LSPs referencing them from other routers on the LAN. By preserving the LAN-ID across the restart, this churn can be prevented. To enable a restarting router to learn the LAN-ID used prior to restart, the LAN-ID specified in an IIH with RR set MUST be ignored.

Transmission of "normal" IIHs is inhibited until the conditions described below are met (in order to avoid causing an unnecessary

adjacency initialization). Upon expiry of the timer T1, it is restarted and the IIH is retransmitted as above.

When a restarting router receives an IIH a local adjacency is established as usual, and if the IIH contains a restart TLV with the RA bit set (and on LAN circuits with a Restart Neighbor System ID that matches that of the local system), the receipt of the acknowledgement over that interface is noted. When the RA bit is set and the state of the remote adjacency is "UP", then the timer T3 is set to the minimum of its current value and the value of the "Remaining Time" field in the received IIH.

On a Point-to-Point link, receipt of an IIH not containing the restart TLV is also treated as an acknowledgement, since it indicates that the neighbor is not restart capable. However, since no CSNP is guaranteed to be received over this interface, the timer T1 is cancelled immediately without waiting for a complete set of CSNPs. Synchronization may therefore be deemed complete even though there are some LSPs which are held (only) by this neighbor (see Section 3.4). In this case, we also want to be certain that the neighbor will reinitialize the adjacency in order to guarantee that the SRMflags have been set on its database, thus ensuring eventual LSPDB synchronization. This is guaranteed to happen except in the case where the Adjacency Three-Way State in the received IIH is "UP" and the Neighbor Extended Local Circuit ID matches the extended local circuit ID assigned by the restarting router. In this case, the restarting router MUST force the adjacency to reinitialize by setting the local Adjacency Three-Way State to "DOWN" and sending a normal IIH.

In the case of a LAN interface, receipt of an IIH not containing the restart TLV is unremarkable since synchronization can still occur so long as at least one of the non-restarting neighboring routers on the LAN supports restart. Therefore, T1 continues to run in this case. If none of the neighbors on the LAN are restart capable, T1 will eventually expire after the locally defined number of retries.

In the case of a Point-to-Point circuit, the "LocalCircuitID" and "Extended Local Circuit ID" information contained in the IIH can be used immediately to generate an IIH containing the correct three-way handshake information. The presence of "Neighbor Extended Local Circuit ID" information that does not match the value currently in use by the local system is ignored (since the IIH may have been transmitted before the neighbor had received the new value from the restarting router), but the adjacency remains in the initializing state until the correct information is received.

In the case of a LAN circuit, the source neighbor information (e.g., SNPAAddress) is recorded and used for adjacency establishment and maintenance as normal.

When BOTH a complete set of CSNPs (for each active level, in the case of a Point-to-Point circuit) and an acknowledgement have been received over the interface, the timer T1 is cancelled.

Once the timer T1 has been cancelled, subsequent IIHs are transmitted according to the normal algorithms, but including the restart TLV with both RR and RA clear.

If a LAN contains a mixture of systems, only some of which support the new algorithm, database synchronization is still guaranteed, but the "old" systems will have reinitialized their adjacencies.

If an interface is active, but does not have any neighboring router reachable over that interface, the timer T1 would never be cancelled, and according to Section 3.4.1.1, the SPF would never be run. Therefore, timer T1 is cancelled after some predetermined number of expirations (which MAY be 1).

2.3.2. Adjacency Acquisition during Start

The starting router wants to ensure that in the event that a neighboring router has an adjacency to the starting router in the "UP" state (from a previous incarnation of the starting router), this adjacency is reinitialized. The starting router also wants neighboring routers to suppress advertisement of an adjacency to the starting router until LSP database synchronization is achieved. This is achieved by sending IIHs with the RR bit clear and the SA bit set in the restart TLV. The RR bit remains clear and the SA bit remains set in subsequent transmissions of IIHs until the adjacency has reached the "UP" state and the initial T1 timer interval (see below) has expired.

Receipt of an IIH with the RR bit clear will result in the neighboring router utilizing normal operation of the adjacency state machine. This will ensure that any old adjacency on the neighboring router will be reinitialized.

Upon receipt of an IIH with the SA bit set, the behavior described in Section 3.2.2 is followed.

Upon starting, a router starts timer T2 for each LSPDB.

For each interface (and in the case of a LAN circuit, for each level), when an adjacency reaches the "UP" state, the starting router

starts a timer T1 and transmits an IIH containing the restart TLV with the RR bit clear and SA bit set. Upon expiry of the timer T1, it is restarted and the IIH is retransmitted with both RR and SA bits set (only the RR bit has changed state from earlier IIHs).

Upon receipt of an IIH with the RR bit set (regardless of whether or not the SA bit is set), the behavior described in Section 2.2.1 is followed.

When an IIH is received by the starting router and the IIH contains a restart TLV with the RA bit set (and on LAN circuits with a Restart Neighbor System ID that matches that of the local system), the receipt of the acknowledgement over that interface is noted.

On a Point-to-Point link, receipt of an IIH not containing the restart TLV is also treated as an acknowledgement, since it indicates that the neighbor is not restart capable. Since the neighbor will have reinitialized the adjacency, this guarantees that SRMflags have been set on its database, thus ensuring eventual LSPDB synchronization. However, since no CSNP is guaranteed to be received over this interface, the timer T1 is cancelled immediately without waiting for a complete set of CSNPs. Synchronization may therefore be deemed complete even though there are some LSPs that are held (only) by this neighbor (see Section 2.4).

In the case of a LAN interface, receipt of an IIH not containing the restart TLV is unremarkable since synchronization can still occur so long as at least one of the non-restarting neighboring routers on the LAN supports restart. Therefore, T1 continues to run in this case. If none of the neighbors on the LAN are restart capable, T1 will eventually expire after the locally defined number of retries. The usual operation of the update process will ensure that synchronization is eventually achieved.

When BOTH a complete set of CSNPs (for each active level, in the case of a Point-to-Point circuit) and an acknowledgement have been received over the interface, the timer T1 is cancelled. Subsequent IIHs sent by the starting router have the RR and RA bits clear and the SA bit set in the restart TLV.

Timer T1 is cancelled after some predetermined number of expirations (which MAY be 1).

When the T2 timer(s) are cancelled or expire, transmission of "normal" IIHs (with RR, RA, and SA bits clear) will begin.

2.3.3. Multiple Levels

A router that is operating as both a Level 1 and a Level 2 router on a particular interface MUST perform the above operations for each level.

On a LAN interface, it MUST send and receive both Level 1 and Level 2 IIHs and perform the CSNP synchronizations independently for each level.

On a Point-to-Point interface, only a single IIH (indicating support for both levels) is required, but it MUST perform the CSNP synchronizations independently for each level.

2.4. Database Synchronization

When a router is started or restarted, it can expect to receive a complete set of CSNPs over each interface. The arrival of the CSNP(s) is now guaranteed, since an IIH with the RR bit set will be retransmitted until the CSNP(s) are correctly received.

The CSNPs describe the set of LSPs that are currently held by each neighbor. Synchronization will be complete when all these LSPs have been received.

When (re)starting, a router starts an instance of timer T2 for each LSPDB as described in Section 3.3.1 or Section 3.3.2. In addition to normal processing of the CSNPs, the set of LSPIDs contained in the first complete set of CSNPs received over each interface is recorded, together with their remaining lifetime. In the case of a LAN interface, a complete set of CSNPs MUST consist of CSNPs received from neighbors that are not restarting. If there are multiple interfaces on the (re)starting router, the recorded set of LSPIDs is the union of those received over each interface. LSPs with a remaining lifetime of zero are NOT so recorded.

As LSPs are received (by the normal operation of the update process) over any interface, the corresponding LSPID entry is removed (it is also removed if an LSP arrives before the CSNP containing the reference). When an LSPID has been held in the list for its indicated remaining lifetime, it is removed from the list. When the list of LSPIDs is empty and the timer T1 has been cancelled for all the interfaces that have an adjacency at this level, the timer T2 is cancelled.

At this point, the local database is guaranteed to contain all the LSP(s) (either the same sequence number or a more recent sequence number) that were present in the neighbors’ databases at the time of

(re)starting. LSPs that arrived in a neighbor's database after the time of (re)starting may or may not be present, but the normal operation of the update process will guarantee that they will eventually be received. At this point, the local database is deemed to be "synchronized".

Since LSPs mentioned in the CSNP(s) with a zero remaining lifetime are not recorded, and those with a short remaining lifetime are deleted from the list when the lifetime expires, cancellation of the timer T2 will not be prevented by waiting for an LSP that will never arrive.

2.4.1. LSP Generation and Flooding and SPF Computation

The operation of a router starting, as opposed to restarting, is somewhat different. These two cases are dealt with separately below.

2.4.1.1. Restarting

In order to avoid causing unnecessary routing churn in other routers, it is highly desirable that the router's own LSPs generated by the restarting system are the same as those previously present in the network (assuming no other changes have taken place). It is important therefore not to regenerate and flood the LSPs until all the adjacencies have been re-established and any information required for propagation into the local LSPs is fully available. Ideally, the information is loaded into the LSPs in a deterministic way, such that the same information occurs in the same place in the same LSP (and hence the LSPs are identical to their previous versions). If this can be achieved, the new versions may not even cause SPF to be run in other systems. However, provided the same information is included in the set of LSPs (albeit in a different order, and possibly different LSPs), the result of running the SPF will be the same and will not cause churn to the forwarding tables.

In the case of a restarting router, none of the router's own LSPs are transmitted, nor are the router's own forwarding tables updated while the timer T3 is running.

Redistribution of inter-level information MUST be regenerated before this router's LSP is flooded to other nodes. Therefore, the Level-n non-pseudonode LSP(s) MUST NOT be flooded until the other level's T2 timer has expired and its SPF has been run. This ensures that any inter-level information that is to be propagated can be included in the Level-n LSP(s).

During this period, if one of the router's own (including pseudonodes) LSPs is received, which the local router does not

currently have in its own database, it is NOT purged. Under normal operation, such an LSP would be purged, since the LSP clearly should not be present in the global LSP database. However, in the present circumstances, this would be highly undesirable, because it could cause premature removal of a router's own LSP -- and hence churn in remote routers. Even if the local system has one or more of the router's own LSPs (which it has generated, but not yet transmitted), it is still not valid to compare the received LSP against this set, since it may be that as a result of propagation between Level 1 and Level 2 (or vice versa), a further router's own LSP will need to be generated when the LSP databases have synchronized.

During this period, a restarting router SHOULD send CSNPs as it normally would. Information about the router's own LSPs MAY be included, but if it is included it MUST be based on LSPs that have been received, not on versions that have been generated (but not yet transmitted). This restriction is necessary to prevent premature removal of an LSP from the global LSP database.

When the timer T2 expires or is cancelled indicating that synchronization for that level is complete, the SPF for that level is run in order to derive any information that is required to be propagated to another level, but the forwarding tables are not yet updated.

Once the other level's SPF has run and any inter-level propagation has been resolved, the router's own LSPs can be generated and flooded. Any own LSPs that were previously ignored, but that are not part of the current set of own LSPs (including pseudonodes), MUST then be purged. Note that it is possible that a Designated Router change may have taken place, and consequently the router SHOULD purge those pseudonode LSPs that it previously owned, but that are now no longer part of its set of pseudonode LSPs.

When all the T2 timers have expired or been cancelled, the timer T3 is cancelled and the local forwarding tables are updated.

If the timer T3 expires before all the T2 timers have expired or been cancelled, this indicates that the synchronization process is taking longer than the minimum holding time of the neighbors. The router's own LSP(s) for levels that have not yet completed their first SPF computation are then flooded with the overload bit set to indicate that the router's LSPDB is not yet synchronized (and therefore other routers MUST NOT compute routes through this router). Normal operation of the update process resumes, and the local forwarding tables are updated. In order to prevent the neighbor's adjacencies from expiring, IIHs with the normal interface value for the holding time are transmitted over all interfaces with neither RR nor RA set

in the restart TLV. This will cause the neighbors to refresh their adjacencies. The router's own LSP(s) will continue to have the overload bit set until timer T2 has expired or been cancelled.

2.4.1.2. Starting

In the case of a starting router, as soon as each adjacency is established, and before any CSNP exchanges, the router's own zeroth LSP is transmitted with the overload bit set. This prevents other routers from computing routes through the router until it has reliably acquired the complete set of LSPs. The overload bit remains set in subsequent transmissions of the zeroth LSP (such as will occur if a previous copy of the router's own zeroth LSP is still present in the network) while any timer T2 is running.

When all the T2 timers have been cancelled, the router's own LSP(s) MAY be regenerated with the overload bit clear (assuming the router is not in fact overloaded, and there is no other reason, such as incomplete BGP convergence, to keep the overload bit set) and flooded as normal.

Other LSPs owned by this router (including pseudonodes) are generated and flooded as normal, irrespective of the timer T2. The SPF is also run as normal and the Routing Information Base (RIB) and Forwarding Information Base (FIB) updated as routes become available.

To avoid the possible formation of temporary blackholes, the starting router sets the SA bit in the restart TLV (as described in Section 3.3.2) in all IIHs that it sends.

When all T2 timers have been cancelled, the starting router MUST transmit IIHs with the SA bit clear.

3. State Tables

This section presents state tables that summarize the behaviors described in this document. Other behaviors, in particular adjacency state transitions and LSP database update operation, are NOT included in the state tables except where this document modifies the behaviors described in [ISO10589] and [RFC5303].

The states named in the columns of the tables below are a mixture of states that are specific to a single adjacency (ADJ suppressed, ADJ Seen RA, ADJ Seen CSNP) and states that are indicative of the state of the protocol instance (Running, Restarting, Starting, SPF Wait).

Three state tables are presented from the point of view of a running router, a restarting router, and a starting router.

3.1. Running Router

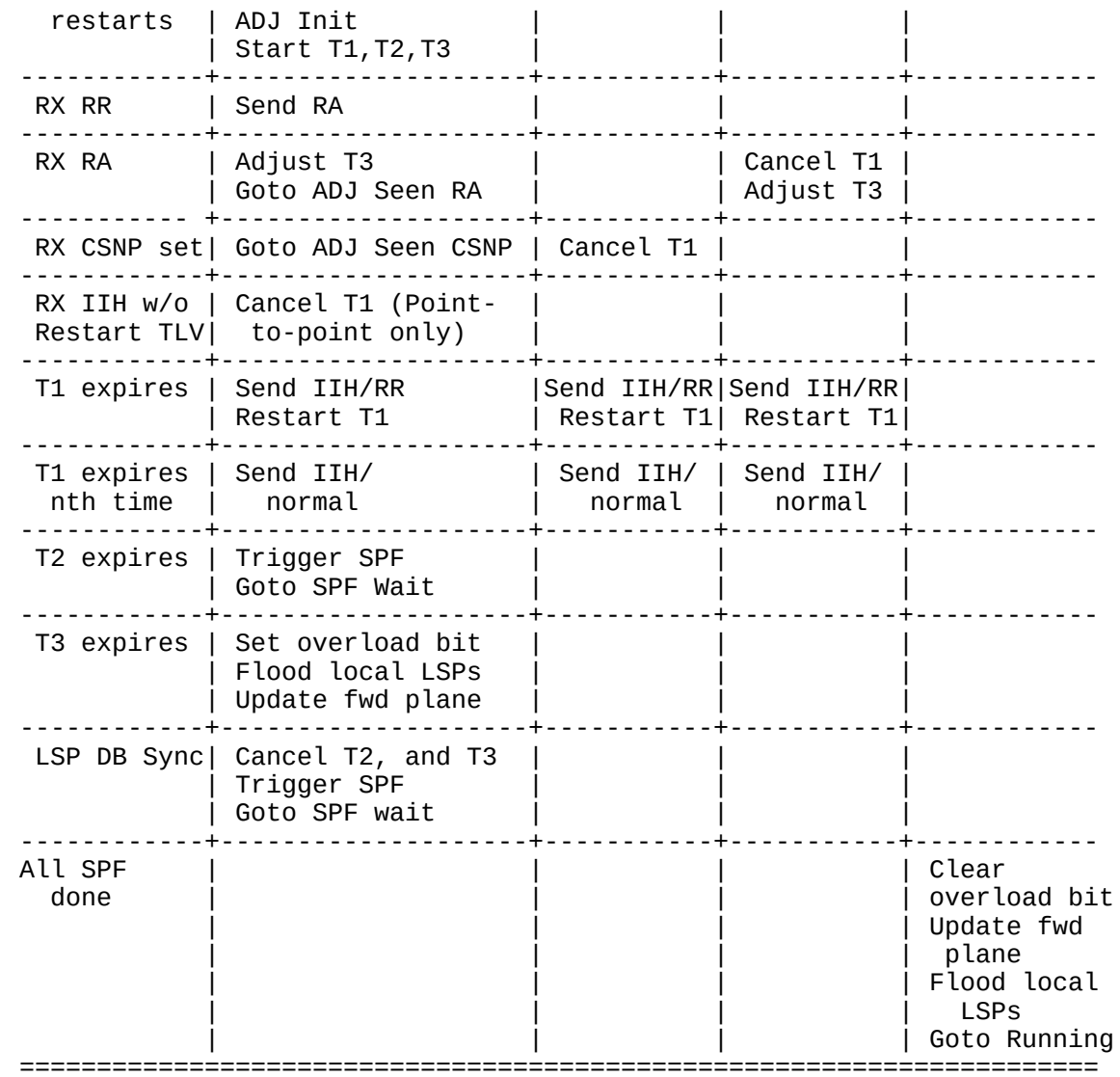
Event	Running	ADJ suppressed
RX PR	Set Planned Restart state. Send PA	
RX PR clr and RR clr	Clear Planned Restart State	
RX RR	Maintain ADJ State Send RA Set SRM,send CSNP (Note 1) Update Hold Time, set Restart Mode (Note 2)	
RX RR clr	Clr Restart mode	
RX SA	Suppress IS neighbor TLV in LSP(s) Goto ADJ Suppressed	
RX SA clr		Unsuppress IS neighbor TLV in LSP(s) Goto Running

Note 1: CSNPs are sent by routers in accordance with Section 2.2.1c

Note 2: If Restart Mode clear

3.2. Restarting Router

Event	Restarting	ADJ Seen RA	ADJ Seen CSNP	SPF Wait
Restart planned	Send PR			
Planned restart canceled	Send PR clr			
Router	Send IIH/RR			



3.3. Starting Router

Event	Starting	ADJ Seen RA	ADJ Seen CSNP
Router starts	Send IIH/SA Start T1,T2		
RX RR	Send RA		
RX RA	Goto ADJ Seen RA		Cancel T1
RX CSNP Set	Goto ADJ Seen CSNP	Cancel T1	
RX IIH w no Restart TLV	Cancel T1 (Point-to-Point only)		
ADJ UP	Start T1 Send local LSPs with overload bit set		
T1 expires	Send IIH/RR and SA Restart T1	Send IIH/RR and SA Restart T1	Send IIH/RR and SA Restart T1
T1 expires nth time	Send IIH/SA	Send IIH/SA	Send IIH/SA
T2 expires	Clear overload bit Send IIH normal Goto Running		
LSP DB Sync	Cancel T2 Clear overload bit Send IIH normal		

4. IANA Considerations

This document defines the following IS-IS TLV that is listed in the IS-IS TLV codepoint registry:

Type	Description	IIH	LSP	SNP
211	Restart TLV	y	n	n

5. Security Considerations

Any new security issues raised by the procedures in this document depend upon the ability of an attacker to inject a false but apparently valid IIH, the ease/difficulty of which has not been altered.

If the RR bit is set in a false IIH, neighbors who receive such an IIH will continue to maintain an existing adjacency in the "UP" state and may (re)send a complete set of CSNPs. While the latter action is wasteful, neither action causes any disruption in correct protocol operation.

If the RA bit is set in a false IIH, a (re)starting router that receives such an IIH may falsely believe that there is a neighbor on the corresponding interface that supports the procedures described in this document. In the absence of receipt of a complete set of CSNPs on that interface, this could delay the completion of (re)start procedures by requiring the timer T1 to time out the locally defined maximum number of retries. This behavior is the same as would occur on a LAN where none of the (re)starting router’s neighbors support the procedures in this document and is covered in Sections 2.3.1 and 2.3.2.

If an SA bit is set in a false IIH, this could cause suppression of the advertisement of an IS neighbor, which could either continue for an indefinite period or occur intermittently with the result being a possible loss of reachability to some destinations in the network and/or increased frequency of LSP flooding and SPF calculation.

The possibility of IS-IS PDU spoofing can be reduced by the use of authentication as described in [RFC1195] and [ISO10589], and especially the use of cryptographic authentication as described in [RFC5304] and [RFC5310].

6. Manageability Considerations

These extensions that have been designed, developed, and deployed for many years do not have any new impact on management and operation of the IS-IS protocol via this standardization process.

7. Acknowledgements

For RFC 5306 the authors acknowledged contributions made by Jeff Parker, Radia Perlman, Mark Schaefer, Naiming Shen, Nischal Sheth, Russ White, and Rena Yang.

The authors of this updated version acknowledge the contribution of Mike Shand, co-auther of RFC 5306.

8. Normative References

[ISO10589] International Organization for Standardization, "Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Second Edition, Nov 2002.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[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>.

[RFC5303] Katz, D., Saluja, R., and D. Eastlake 3rd, "Three-Way Handshake for IS-IS Point-to-Point Adjacencies", RFC 5303, DOI 10.17487/RFC5303, October 2008, <https://www.rfc-editor.org/info/rfc5303>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[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>.

Appendix A. Summary of Changes from RFC 5306

This document extends RFC 5306 by introducing support for signalling the neighbors of a restarting router that a planned restart is about to occur. This allows the neighbors to be aware of the state of the restarting router so that appropriate action may be taken if other topology changes occur while the planned restart is in progress. Since the forwarding plane of the restarting router is maintained based upon the pre-restart state of the network, additional topology changes introduce the possibility that traffic may be lost if paths via the restarting router continue to be used while the restart is in progress.

In support of this new functionality two new flags have been introduced:

PR - Restart is planned
PA - Planned restart acknowledgement

No changes to the post restart exchange between the restarting router and its neighbors have been introduced.

Authors' Addresses

Les Ginsberg
Cisco Systems, Inc.

Email: ginsberg@cisco.com

Paul Wells
Cisco Systems, Inc.

Email: pauwells@cisco.com

IS-IS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2018

S. Litkowski
Orange
Y. Qu
Huawei
P. Sarkar
Individual
I. Chen
Jabil
J. Tantsura
Individual
June 29, 2018

YANG Data Model for IS-IS Segment Routing
draft-ietf-isis-sr-yang-04

Abstract

This document defines a YANG data model that can be used to configure and manage IS-IS Segment Routing ([I-D.ietf-isis-segment-routing-extensions].

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].

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 31, 2018.

Table of Contents

1. Overview

YANG [RFC6020] [RFC7950] is a data definition language used to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces

(e.g., ReST) and encodings other than XML (e.g., JSON) are being defined. Furthermore, YANG data models can be used as the basis for implementation of other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage IS-IS Segment Routing and it is an augmentation to the IS-IS YANG data model.

2. IS-IS Segment Routing

This document defines a model for IS-IS Segment Routing feature. It is an augmentation of the IS-IS base model.

The IS-IS SR YANG module requires support for the base segment routing module [I-D.ietf-spring-sr-yang], which defines the global segment routing configuration independent of any specific routing protocol configuration, and support of IS-IS base model [I-D.ietf-isis-yang-isis-cfg] which defines basic IS-IS configuration and state.

The figure below describes the overall structure of the isis-sr YANG module:

```
 module: ietf-isis-sr
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis:
     +--rw segment-routing
     |  +--rw enabled?    boolean
     |  +--rw bindings
     |     +--rw advertise
     |     |  +--rw policies*   string
     |     +--rw receive?     boolean
     +--rw protocol-srgb {sr:protocol-srgb}?
        +--rw srgb* [lower-bound upper-bound]
           +--rw lower-bound    uint32
           +--rw upper-bound    uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:interfaces
           /isis:interface:
     +--rw segment-routing
        +--rw adjacency-sid
           +--rw advertise-adj-group-sid* [group-id]
           |  +--rw group-id    uint32
           +--rw advertise-protection?      enumeration
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:interfaces
           /isis:interface/isis:fast-reroute:
```

```
     +--rw ti-lfa {ti-lfa}?
        +--rw enable?   boolean
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:interfaces
           /isis:interface/isis:fast-reroute/isis:lfa/isis:remote-lfa:
     +--rw use-segment-routing-path?   boolean {remote-lfa-sr}?
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:interfaces
           /isis:interface/isis:adjacencies/isis:adjacency:
     +--ro adjacency-sid* [value]
        +--ro af?                     iana-rt-types:address-family
        +--ro value                   uint32
        +--ro weight?                 uint8
        +--ro protection-requested?   boolean
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:router-capabilities:
     +--ro sr-capability
     |  +--ro flags?           bits
     |  +--ro global-blocks
     |     +--ro global-block*
     |        +--ro range-size?    uint32
     |        +--ro sid-sub-tlv
     |           +--ro sid?   uint32
     +--ro sr-algorithms
     |  +--ro sr-algorithm*   uint8
     +--ro local-blocks
     |  +--ro local-block*
     |     +--ro range-size?    uint32
     |     +--ro sid-sub-tlv
     |        +--ro sid?   uint32
     +--ro srms-preference
        +--ro preference?   uint8
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:extended-is-neighbor
           /isis:neighbor:
     +--ro sid-list* [value]
        +--ro flags?         bits
        +--ro weight?        uint8
        +--ro neighbor-id?   isis:system-id
        +--ro value          uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:mt-is-neighbor/isis:neighbor:
     +--ro sid-list* [value]
        +--ro flags?         bits
        +--ro weight?        uint8
```

```
        +--ro neighbor-id?   isis:system-id
        +--ro value          uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:extended-ipv4-reachability
           /isis:prefixes:
     +--ro sid-list* [value]
        +--ro flags?       bits
        +--ro algorithm?   uint8
        +--ro value        uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:mt-extended-ipv4-reachability
           /isis:prefixes:
     +--ro sid-list* [value]
        +--ro flags?       bits
        +--ro algorithm?   uint8
        +--ro value        uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:ipv6-reachability/isis:prefixes:
     +--ro sid-list* [value]
        +--ro flags?       bits
        +--ro algorithm?   uint8
        +--ro value        uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp/isis:mt-ipv6-reachability
           /isis:prefixes:
     +--ro sid-list* [value]
        +--ro flags?       bits
        +--ro algorithm?   uint8
        +--ro value        uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/isis:isis/isis:database
           /isis:level-db/isis:lsp:
     +--ro segment-routing-bindings* [fec range]
        +--ro fec        string
        +--ro range      uint16
        +--ro flags?     bits
        +--ro binding
           +--ro prefix-sid
              +--ro sid-list* [value]
                 +--ro flags?       bits
                 +--ro algorithm?   uint8
                 +--ro value        uint32
```

3. IS-IS Segment Routing configuration

3.1. Segment Routing activation

Activation of segment-routing IS-IS is done by setting the "enable" leaf to true. This triggers advertisement of segment-routing extensions based on the configuration parameters that have been setup using the base segment routing module.

3.2. Advertising mapping server policy

The base segment routing module defines mapping server policies. By default, IS-IS will not advertise nor receive any mapping server entry. The IS-IS segment-routing module allows to advertise one or multiple mapping server policies through the "bindings/advertise/ policies" leaf-list. The "bindings/receive" leaf allows to enable the reception of mapping server entries.

3.3. IP Fast reroute

IS-IS SR model augments the fast-reroute container under interface. It brings the ability to activate TI-LFA (topology independent LFA) and also enhances remote LFA to use segment-routing tunneling instead of LDP.

4. IS-IS Segment Routing YANG Module

```
<CODE BEGINS> file "ietf-isis-sr@2018-06-25.yang"
module ietf-isis-sr {
  namespace "urn:ietf:params:xml:ns:"
          + "yang:ietf-isis-sr";
  prefix isis-sr;


  import ietf-routing {
    prefix "rt";
  }

  import ietf-segment-routing-common {
  prefix "sr-cmn";
  }


  import ietf-segment-routing {
  prefix "sr";
  }

  import ietf-isis {
```

```
    prefix "isis";
    }

    import iana-routing-types {
      prefix "iana-rt-types";
    }

    organization
     "IETF LSR - LSR Working Group";

    contact
      "WG List:  <mailto:lsr@ietf.org>

      Editor:    Stephane Litkowski
                 <mailto:stephane.litkowski@orange.com>

      Author:    Acee Lindem
                 <mailto:acee@cisco.com>
      Author:    Yingzhen Qu
                 <mailto:yiqu@cisco.com>
      Author:    Pushpasis Sarkar
                 <mailto:pushpasis.ietf@gmail.com>
      Author:    Ing-Wher Chen
                 <mailto:ichen@kuatrotech.com>
      Author:    Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>
      ";

    description
      "The YANG module defines a generic configuration model for
      Segment routing ISIS extensions common across all of the vendor
      implementations.";

    revision 2018-06-25 {
    description
      "Initial revision.";
    reference "RFC XXXX";
    }


    /* Identities */


    /* Features */

    feature remote-lfa-sr {
      description
       "Enhance rLFA to use SR path.";
```

```
    }

    feature ti-lfa {
      description
       "Enhance IPFRR with ti-lfa
       support";
    }

    /* Groupings */


    grouping sid-sub-tlv {
      description "SID/Label sub-TLV grouping.";
      container sid-sub-tlv {
        description
          "Used to advertise the SID/Label associated with a
        prefix or adjacency.";
        leaf sid {
          type uint32;
        description
          "Segment Identifier (SID) - A 20 bit label or
           32 bit SID.";
        }
      }
    }

    grouping sr-capability {
      description
        "SR capability grouping.";
      container sr-capability {
        description
          "Segment Routing capability.";
        leaf flags {
          type bits {
            bit mpls-ipv4 {
              position 0;
              description
                "If set, then the router is capable of
                 processing SR MPLS encapsulated IPv4 packets
                 on all interfaces.";
            }
            bit mpls-ipv6 {
              position 1;
              description
                "If set, then the router is capable of
                 processing SR MPLS encapsulated IPv6 packets
                 on all interfaces.";
            }
```

```
          }
          description
            "Flags.";
        }
        container global-blocks {
          description
            "Segment Routing Global Blocks.";
          list global-block {
            description "Segment Routing Global Block.";
            leaf range-size {
              type uint32;
              description "The SID range.";
            }
            uses sid-sub-tlv;
          }
        }
      }
    }

    grouping sr-algorithm {
      description
        "SR algorithm grouping.";
      container sr-algorithms {
        description "All SR algorithms.";
        leaf-list sr-algorithm {
          type uint8;
          description
            "The Segment Routing (SR) algorithms that the router is
             currently using.";
        }
      }
    }

    grouping srlb {
      description
        "SR Local Block grouping.";
      container local-blocks {
        description "List of SRLBs.";
        list local-block {
          description "Segment Routing Local Block.";
          leaf range-size {
            type uint32;
            description "The SID range.";
          }
          uses sid-sub-tlv;
        }
      }
    }
```

```
    grouping srms-preference {
      description "The SRMS preference TLV is used to advertise
                   a preference associated with the node that acts
                   as an SR Mapping Server.";
      container srms-preference {
        description "SRMS Preference TLV.";
        leaf preference {
          type uint8 {
            range "0 .. 255";
          }
          description "SRMS preference TLV, vlaue from 0 to 255.";
        }
      }
    }

    grouping adjacency-state {
      description
        "This group will extend adjacency state.";
      list adjacency-sid {
        key value;
        config false;
        leaf af {
          type iana-rt-types:address-family;
          description
            "Address-family associated with the
             segment ID";
        }
        leaf value {
          type uint32;
          description
            "Value of the Adj-SID.";
        }
        leaf weight {
          type uint8;
          description
            "Weight associated with
            the adjacency SID.";
        }
        leaf protection-requested {
          type boolean;
          description
            "Describe if the adjacency SID
             must be protected.";
        }
        description
          "List of adjacency Segment IDs.";
      }
    }
```

```
    grouping prefix-segment-id {
      description
        "This group defines segment routing extensions
         for prefixes.";

      list sid-list {
        key value;

        leaf flags {
          type bits {
            bit readvertisment {
              position 7;
              description
                "If set, then the prefix to
                 which this Prefix-SID is attached,
                 has been propagated by the
                 router either from another level
                 or from redistribution.";
            }
            bit php {
              position 5;
              description
                "If set, then the penultimate hop MUST NOT
                 pop the Prefix-SID before delivering the packet
                 to the node that advertised the Prefix-SID.";
            }
            bit explicit-null {
              position 4;
              description
                "If set, any upstream neighbor of
                 the Prefix-SID originator MUST replace
                 the Prefix-SID with a
                 Prefix-SID having an
                 Explicit-NULL value (0 for IPv4 and 2 for
                 IPv6) before forwarding the packet.";
            }
            bit value {
              position 3;
              description
                "If set, then the Prefix-SID carries a
                 value (instead of an index).
                 By default the flag is UNSET.";
            }
            bit local {
              position 2;
              description
                "If set, then the value/index carried by
                 the Prefix-SID has local significance.
```

```
                By default the flag is UNSET.";
            }
          }
          description
            "Describes flags associated with the
             segment ID.";
        }

        leaf algorithm {
          type uint8;
          description
            "Algorithm to be used for path computation.";
        }
        leaf value {
          type uint32;
          description
           "Value of the prefix-SID.";
        }
        description
          "List of segments.";
      }
    }

    grouping adjacency-segment-id {
      description
        "This group defines segment routing extensions
         for adjacencies.";

      list sid-list {
        key value;

        leaf flags {
          type bits {
            bit address-family {
              position 7;
              description
                "If unset, then the Adj-SID refers
                 to an adjacency with outgoing IPv4 encapsulation.
                 If set then the Adj-SID refers to an adjacency
                 with outgoing IPv6 encapsulation.";
            }
            bit backup {
              position 6;
              description
                "If set, the Adj-SID refers to an
                 adjacency being protected
                 (e.g.: using IPFRR or MPLS-FRR)";
            }
```

```
            bit value {
              position 5;
              description
                "If set, then the SID carries a
                 value (instead of an index).
                 By default the flag is SET.";
            }
            bit local {
              position 4;
              description
                "If set, then the value/index carried by
                 the SID has local significance.
                 By default the flag is SET.";
            }
            bit set {
              position 3;
              description
                "When set, the S-Flag indicates that the
                 Adj-SID refers to a set of adjacencies.";
            }
            bit persistent {
              position 2;
              description
                "When set, the P-Flag indicates that the
                 Adj-SID is persistently allocated.";
            }
          }

          description
            "Describes flags associated with the
             segment ID.";
        }
        leaf weight {
          type uint8;
          description
            "The value represents the weight of the Adj-SID
             for the purpose of load balancing.";
        }
        leaf neighbor-id {
          type isis:system-id;
          description
            "Describes the system ID of the neighbor
             associated with the SID value. This is only
             used on LAN adjacencies.";
        }
        leaf value {
          type uint32;
          description
```

```
            "Value of the Adj-SID.";
        }
        description
          "List of segments.";
      }
    }

    grouping segment-routing-binding-tlv {
      list segment-routing-bindings {
        key "fec range";

        leaf fec {
          type string;
          description
          "IP (v4 or v6) range to be bound to SIDs.";
        }

        leaf range {
          type uint16;
          description
            "Describes number of elements to assign
             a binding to.";
        }

        leaf flags {
          type bits {
            bit address-family {
              position 7;
              description
                "If unset, then the Prefix FEC
                 carries an IPv4 Prefix.
                 If set then the Prefix FEC carries an
                 IPv6 Prefix.";
            }
            bit mirror {
              position 6;
              description
                "Set if the advertised SID/path
                 corresponds to a mirrored context.";
            }
            bit flooding {
              position 5;
              description
                "If the S bit is set(1),
                 the IS-IS Router CAPABILITY TLV
                 MUST be flooded across the entire routing domain.
                 If the S bit is
                 not set(0), the TLV MUST NOT be leaked between levels.
```

```
                This bit MUST NOT be altered during the TLV leaking.";
            }
            bit down {
              position 4;
              description
                "When the IS-IS Router CAPABILITY TLV is
                 leaked from level-2 to level-1, the D bit
                 MUST be set.  Otherwise, this bit MUST
                 be clear.  IS-IS Router capability TLVs
                 with the D bit set MUST NOT
                 be leaked from level-1 to level-2.
                 This is to prevent TLV looping.";
            }
            bit attached {
              position 3;
              description
                "The originator of the SID/Label Binding TLV MAY set the
                 A bit in order to signal that the prefixes and SIDs
                 advertised in the SID/Label Binding TLV are directly
                 connected to their originators.";
            }

          }
          description
            "Flags of the binding.";
        }

        container binding {
          container prefix-sid {
            uses prefix-segment-id;
            description
              "Binding prefix SID to the range.";
          }
          description
            "Bindings associated with the range.";
        }

        description
          "This container describes list of SID/Label bindings.
           ISIS reference is TLV 149.";
      }
      description
        "Defines binding TLV for database.";
    }

    /* Cfg */

    augment "/rt:routing/" +
```

```
           "rt:control-plane-protocols/rt:control-plane-protocol"+
           "/isis:isis" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol configuration
         with segment routing.";

      uses sr:controlplane-cfg;
      container protocol-srgb {
        if-feature sr:protocol-srgb;
        uses sr-cmn:srgb-cfg;
        description
          "Per-protocol SRGB.";
      }
    }

    augment "/rt:routing/" +
           "rt:control-plane-protocols/rt:control-plane-protocol"+
           "/isis:isis/isis:interfaces/isis:interface" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol configuration
         with segment routing.";

      uses sr:igp-interface-cfg;
    }

    augment "/rt:routing/" +
           "rt:control-plane-protocols/rt:control-plane-protocol"+
           "/isis:isis/isis:interfaces/isis:interface"+
           "/isis:fast-reroute" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS IP FRR with TILFA.";

      container ti-lfa {
```

```
        if-feature ti-lfa;
        leaf enable {
          type boolean;
          description
            "Enables TI-LFA computation.";
        }
        description
          "TILFA configuration.";
      }
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:interfaces/isis:interface"+
            "/isis:fast-reroute/isis:lfa/isis:remote-lfa" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS remoteLFA config with
         use of segment-routing path.";

      leaf use-segment-routing-path {
        if-feature remote-lfa-sr;
        type boolean;
        description
          "force remote LFA to use segment routing
           path instead of LDP path.";
      }
    }

    /* Operational states */

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:interfaces/isis:interface" +
            "/isis:adjacencies/isis:adjacency" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol configuration
         with segment routing.";
```

```
      uses adjacency-state;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:router-capabilities" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB router capability.";

      uses sr-capability;
      uses sr-algorithm;
      uses srlb;
      uses srms-preference;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:extended-is-neighbor/isis:neighbor" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB neighbor.";
         uses adjacency-segment-id;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:mt-is-neighbor/isis:neighbor" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB neighbor.";
         uses adjacency-segment-id;
    }
```

```
    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:extended-ipv4-reachability/isis:prefixes" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB prefix.";
         uses prefix-segment-id;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:mt-extended-ipv4-reachability/isis:prefixes" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB prefix.";
      uses prefix-segment-id;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:ipv6-reachability/isis:prefixes" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB prefix.";
      uses prefix-segment-id;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp"+
            "/isis:mt-ipv6-reachability/isis:prefixes" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
```

```
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB prefix.";
      uses prefix-segment-id;
    }

    augment "/rt:routing/" +
            "rt:control-plane-protocols/rt:control-plane-protocol"+
            "/isis:isis/isis:database/isis:level-db/isis:lsp" {
      when "/rt:routing/rt:control-plane-protocols/"+
           "rt:control-plane-protocol/rt:type = 'isis:isis'" {
        description
          "This augment ISIS routing protocol when used";
      }
      description
        "This augments ISIS protocol LSDB.";
      uses segment-routing-binding-tlv;
    }


    /* Notifications */

  }
  <CODE ENDS>
```

5. Security Considerations

Configuration and state data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241].

As IS-IS is an IGP protocol (critical piece of the network), ensuring stability and security of the protocol is mandatory for the network service.

Authors recommends to implement NETCONF access control model ([RFC6536]) to restrict access to all or part of the configuration to specific users.

6. Contributors

Authors would like to thank Derek Yeung, Acee Lindem, Yi Yang for their major contributions to the draft.

7. Acknowledgements

TBD.

8. IANA Considerations

The IANA is requested to assign two new URIs from the IETF XML registry ([RFC3688]). Authors are suggesting the following URI:

```
URI: urn:ietf:params:xml:ns:yang:ietf-isis-sr
Registrant Contact: IS-IS WG
XML: N/A, the requested URI is an XML namespace
```

This document also requests one new YANG module name in the YANG Module Names registry ([RFC6020]) with the following suggestion :

```
name: ietf-isis-sr
namespace: urn:ietf:params:xml:ns:yang:ietf-isis-sr
prefix: isis-sr
reference: RFC XXXX
```

9. Change log for ietf-isis-sr YANG module

9.1. From version -03 to version -04

- Fixed yang module indentations.

9.2. From version -02 to version -03

- Change address-family type according to routing types.

9.3. From isis-sr document version -01 to version -02

- NMDA compliancy.
- Added SRLB in configuration and LSDB.
- Added SR capability in LSDB.
- Added SR algorithms in LSDB.
- Added SRMS preference in LSDB.
- Alignment with iana-rt-types module.
- Align binding SID with draft-ietf-isis-segment-routing-extensions-13.

9.4. From isis-sr document version -00 to version -01

- Added P-Flag in Adj-SID.

9.5. From isis document version -12 to isis-sr document version -00

- Separate document for IS-IS SR extensions.

9.6. From isis document version -12 to version -13

- Align with new segment routing common module.

9.7. From isis document version -09 to version -11

- Fixed XPATH in 'when' expressions.

9.8. From isis document version -08 to version -09

- Align to draft-ietf-netmod-routing-cfg-23.

9.9. From isis document version -07 to version -08

- Align to draft-ietf-netmod-routing-cfg-21.

10. Normative References

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-15 (work in progress), December 2017.

[I-D.ietf-isis-yang-isis-cfg]
Litkowski, S., Yeung, D., Lindem, A., Zhang, Z., and L. Lhotka, "YANG Data Model for IS-IS protocol", draft-ietf-isis-yang-isis-cfg-19 (work in progress), November 2017.

[I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-08 (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>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[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>.

Authors' Addresses

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Yinghzen Qu
Huawei

Email: yingzhen.qu@huawei.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Ing-Wher Chen
Jabil

Email: ing-wher_chen@jabil.com

Jeff Tantsura
Individual

Email: jefftant.ietf@gmail.com

IS-IS Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 10, 2019

S. Litkowski
Orange
D. Yeung
Arrcus, Inc
A. Lindem
Cisco Systems
J. Zhang
Juniper Networks
L. Lhotka
CZ.NIC
August 09, 2018

YANG Data Model for IS-IS protocol
draft-ietf-isis-yang-isis-cfg-24

Abstract

This document defines a YANG data model that can be used to configure and manage IS-IS protocol on network elements.

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].

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 February 10, 2019.

Table of Contents

1. Introduction

This document defines a YANG data model for IS-IS routing protocol.

The data model covers configuration of an IS-IS routing protocol instance as well as operational states.

1.1. Tree diagram

A simplified graphical representation of the data model is presented in Section 2.

The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of the Data Model

The IS-IS YANG module augments the "control-plane-protocol" list in ietf-routing module with specific IS-IS parameters.

The module is designed as per NMDA (Network Management Datastore Architecture).

The figure below describes the overall structure of the isis YANG module:

```
module: ietf-isis
  augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route:
    +--ro metric?       uint32
    +--ro tag*          uint64
    +--ro route-type?   enumeration
  augment /if:interfaces/if:interface:
    +--rw clns-mtu?   uint16
  augment /rt:routing/rt:control-plane-protocols/
    |      rt:control-plane-protocol:
    +--rw isis
       +--rw enable?                    boolean {admin-control}?
       +--rw level-type?                level
       +--rw system-id?                 system-id
       +--rw maximum-area-addresses?    uint8 {maximum-area-addresses}?
       +--rw area-address*              area-address
       +--rw mpls
       |  +--rw te-rid {te-rid}?
       |  |     ...
       |  +--rw ldp
       |        ...
       +--rw auto-cost {auto-cost}?
       |  +--rw reference-bandwidth?   uint32
       |  +--rw enable?                boolean
       +--rw lsp-mtu?                  uint16
       +--rw lsp-lifetime?             uint16
       +--rw lsp-refresh?
       |      rt-types:timer-value-seconds16 {lsp-refresh}?
       +--rw graceful-restart {graceful-restart}?
       |  +--rw enable?             boolean
       |  +--rw restart-interval?   rt-types:timer-value-seconds16
       |  +--rw helper-enable?      boolean
       +--rw nsr {nsr}?
       |  +--rw enable?   boolean
       +--rw node-tags {node-tag}?
       |  +--rw node-tag* [tag]
       |        ...
```

```
        +--rw authentication
        |  +--rw (authentication-type)?
        |  |     ...
        |  +--rw level-1
        |  |     ...
        |  +--rw level-2
        |        ...
        +--rw metric-type
        |  +--rw value?     enumeration
        |  +--rw level-1
        |  |     ...
        |  +--rw level-2
        |        ...
        +--rw default-metric
        |  +--rw value?     wide-metric
        |  +--rw level-1
        |  |     ...
        |  +--rw level-2
        |        ...
        +--rw afs {nlpid-control}?
        |  +--rw af* [af]
        |        ...
        +--rw preference
        |  +--rw (granularity)?
        |        ...
        +--rw overload
        |  +--rw status?   boolean
        +--rw overload-max-metric {overload-max-metric}?
        |  +--rw timeout?   rt-types:timer-value-seconds16
        +--rw fast-reroute {fast-reroute}?
        |  +--rw lfa {lfa}?
        |  +--ro protected-routes
        |  |     ...
        |  +--ro unprotected-routes
        |  |     ...
        |  +--ro protection-statistics* [frr-protection-method]
        |        ...
        +--rw spf-control
        |  +--rw paths?            uint16 {max-ecmp}?
        |  +--rw ietf-spf-delay {ietf-spf-delay}?
        |        ...
        +--rw topologies {multi-topology}?
        |  +--rw topology* [name]
        |        ...
        +--rw interfaces
        |  +--rw interface* [name]
        |        ...
        +--ro spf-log
```

```
         |  +--ro event* [id]
         |        ...
         +--ro lsp-log
         |  +--ro event* [id]
         |        ...
         +--ro hostnames
         |  +--ro hostname* [system-id]
         |        ...
         +--ro database
         |  +--ro level-db* [level]
         |        ...
         +--ro local-rib
         |  +--ro route* [prefix]
         |        ...
         +--ro system-counters
            +--ro level* [level]
                  ...

    rpcs:
      +---x clear-adjacency
      |  +---w input
      |     +---w routing-protocol-instance-name    instance-state-ref
      |     +---w level?                            level
      |     +---w interface?                        string
      +---x clear-database
         +---w input
            +---w routing-protocol-instance-name    instance-state-ref
            +---w level?                            level

    notifications:
      +---n database-overload
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro overload?                enumeration
      +---n lsp-too-large
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro pdu-size?                uint32
      |  +--ro lsp-id?                  lsp-id
      +---n if-state-change
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
```

```
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro state?                   if-state-type
      +---n corrupted-lsp-detected
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro lsp-id?                  lsp-id
      +---n attempt-to-exceed-max-sequence
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro lsp-id?                  lsp-id
      +---n id-len-mismatch
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro pdu-field-len?           uint8
      |  +--ro raw-pdu?                 binary
      +---n max-area-addresses-mismatch
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro max-area-addresses?      uint8
      |  +--ro raw-pdu?                 binary
      +---n own-lsp-purge
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro lsp-id?                  lsp-id
      +---n sequence-number-skipped
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro lsp-id?                  lsp-id
```

```
      +---n authentication-type-failure
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro raw-pdu?                 binary
      +---n authentication-failure
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro raw-pdu?                 binary
      +---n version-skew
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro protocol-version?        uint8
      |  +--ro raw-pdu?                 binary
      +---n area-mismatch
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro raw-pdu?                 binary
      +---n rejected-adjacency
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro raw-pdu?                 binary
      |  +--ro reason?                  string
      +---n protocols-supported-mismatch
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
```

```
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro raw-pdu?                 binary
      |  +--ro protocols*               uint8
      +---n lsp-error-detected
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro lsp-id?                  lsp-id
      |  +--ro raw-pdu?                 binary
      |  +--ro error-offset?            uint32
      |  +--ro tlv-type?                uint8
      +---n adjacency-state-change
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro neighbor?                string
      |  +--ro neighbor-system-id?      system-id
      |  +--ro state?                   adj-state-type
      |  +--ro reason?                  string
      +---n lsp-received
      |  +--ro routing-instance?        string
      |  +--ro routing-protocol-name?   string
      |  +--ro isis-level?              level
      |  +--ro interface-name?          string
      |  +--ro interface-level?         level
      |  +--ro extended-circuit-id?     extended-circuit-id
      |  +--ro lsp-id?                  lsp-id
      |  +--ro sequence?                uint32
      |  +--ro received-timestamp?      yang:timestamp
      |  +--ro neighbor-system-id?      system-id
      +---n lsp-generation
         +--ro routing-instance?        string
         +--ro routing-protocol-name?   string
         +--ro isis-level?              level
         +--ro lsp-id?                  lsp-id
         +--ro sequence?                uint32
         +--ro send-timestamp?          yang:timestamp
```

2.1. IS-IS Configuration

The IS-IS configuration container is divided in:

- Global parameters.
- Per interface configuration (see Section 2.4).

Additional modules may be created this to support any additional parameters. These additional modules should augment the ietf-isis module.

The model implements features, thus some of the configuration statement becomes optional. As an example, the ability to control the administrative state of a particular IS-IS instance is optional. By advertising the feature "admin-control", a device communicates to the client that it supports the ability to shutdown a particular IS-IS instance.

The global configuration contains usual IS-IS parameters such as lsp-mtu, lsp-lifetime, lsp-refresh, default-metric...

2.2. Multitopology Parameters

The model supports multitopology (MT) IS-IS as defined in [RFC5120].

The "topologies" container is used to enable support of MT extensions.

The "name" used in the topology list should refer to an existing RIB of the device.

Some specific parameters could be defined on a per topology basis both at global level and at interface level: for example, an interface metric can be defined per topology.

Multiple address families (like IPv4 or IPv6) can also be activated within the default topology. This can be achieved using the "afs" container (requiring "nlpid-control" feature to be advertised).

2.3. Per-Level Parameters

Some parameters allow a per level configuration. In this case, the parameter is modeled as a container with three configuration locations:

- a top level container: corresponds to level-1-2, so the configuration applies to both levels.

o a level-1 container: corresponds to level-1 specific parameters.

o a level-2 container: corresponds to level-2 specific parameters.

```
            +--rw priority
            |  +--rw value?     uint8
            |  +--rw level-1
            |  |  +--rw value?   uint8
            |  +--rw level-2
            |     +--rw value?   uint8
```

Example:

```
        <priority>
            <value>250</value>
            <level-1>
                <value>100</value>
            </level-1>
            <level-2>
                <value>200</value>
            </level-2>
        </priority>
```

An implementation SHOULD prefer a level specific parameter over a level-all parameter. As example, if the priority is 100 for the level-1, 200 for the level-2 and 250 for the top level configuration, the implementation should use 100 for the level-1 and 200 for the level-2.

Some parameters like "overload bit" and "route preference" are not modeled to support a per level configuration. If an implementation supports per level configuration for such parameter, this implementation SHOULD augment the current model by adding both level-1 and level-2 containers and SHOULD reuse existing configuration groupings.

Example of augmentation:

```
   augment "/rt:routing/" +
           "rt:control-plane-protocols/rt:control-plane-protocol"+
           "/isis:isis/isis:overload" {
               when "rt:type = 'isis:isis'" {
         description
         "This augment IS-IS routing protocol when used";
       }
       description
        "This augments IS-IS overload configuration
         with per level configuration.";

           container level-1 {
                   uses isis:overload-global-cfg;
                   description
                    "Level 1 configuration.";
           }
           container level-2 {
                   uses isis:overload-global-cfg;
                   description
                    "Level 2 configuration.";
           }

   }
```

If an implementation does not support per level configuration for a parameter modeled with per level configuration, the implementation SHOULD advertise a deviation to announce the non support of the level-1 and level-2 containers.

Finally, if an implementation supports per level configuration but does not support the level-1-2 configuration, it SHOULD also advertise a deviation.

2.4. Per-Interface Parameters

The per-interface section of the IS-IS instance describes the interface specific parameters.

The interface is modeled as a reference to an existing interface defined in the "ietf-interfaces" YANG model.

Each interface has some interface-specific parameters that may have a different per level value as described in previous section. An interface-specific parameter always override an IS-IS global parameter.

Some parameters like hello-padding are defined as containers to allow easy extension by vendor specific modules.

```
      +--rw interfaces
      |  +--rw interface* [name]
      |     +--rw name                        if:interface-ref
      |     +--rw level-type?                 level
      |     +--rw lsp-pacing-interval?
      |     |       rt-types:timer-value-milliseconds
      |     +--rw lsp-retransmit-interval?
      |     |       rt-types:timer-value-seconds16
      |     +--rw passive?                    boolean
      |     +--rw csnp-interval?
      |     |       rt-types:timer-value-seconds16
      |     +--rw hello-padding
      |     |  +--rw enable?   boolean
      |     +--rw mesh-group-enable?          mesh-group-state
      |     +--rw mesh-group?                 uint8
      |     +--rw interface-type?             interface-type
      |     +--rw enable?                     boolean {admin-control}?
      |     +--rw tag*                        uint32 {prefix-tag}?
      |     +--rw tag64*                      uint64 {prefix-tag64}?
      |     +--rw node-flag?                  boolean {node-flag}?
      |     +--rw hello-authentication
      |     |  +--rw (authentication-type)?
      |     |  |  +--:(key-chain) {key-chain}?
      |     |  |  |  +--rw key-chain?          key-chain:key-chain-ref
      |     |  |  +--:(password)
      |     |  |     +--rw key?                string
      |     |  |     +--rw crypto-algorithm?   identityref
      |     |  +--rw level-1
      |     |  |  +--rw (authentication-type)?
      |     |  |     +--:(key-chain) {key-chain}?
      |     |  |     |  +--rw key-chain?       key-chain:key-chain-ref
      |     |  |     +--:(password)
      |     |  |        +--rw key?                string
      |     |  |        +--rw crypto-algorithm?   identityref
      |     |  +--rw level-2
      |     |     +--rw (authentication-type)?
      |     |        +--:(key-chain) {key-chain}?
      |     |        |  +--rw key-chain?       key-chain:key-chain-ref
      |     |        +--:(password)
      |     |           +--rw key?                string
      |     |           +--rw crypto-algorithm?   identityref
      |     +--rw hello-interval
      |     |  +--rw value?     rt-types:timer-value-seconds16
      |     |  +--rw level-1
      |     |  |  +--rw value?   rt-types:timer-value-seconds16
      |     |  +--rw level-2
      |     |     +--rw value?   rt-types:timer-value-seconds16
      |     +--rw hello-multiplier
```

```
|     |  +--rw value?     uint16
|     |  +--rw level-1
|     |  |  +--rw value?   uint16
|     |  +--rw level-2
|     |     +--rw value?   uint16
|     +--rw priority
|     |  +--rw value?     uint8
|     |  +--rw level-1
|     |  |  +--rw value?   uint8
|     |  +--rw level-2
|     |     +--rw value?   uint8
|     +--rw metric
|     |  +--rw value?     wide-metric
|     |  +--rw level-1
|     |  |  +--rw value?   wide-metric
|     |  +--rw level-2
|     |     +--rw value?   wide-metric
|     +--rw bfd {bfd}?
|     |  +--rw enable?                              boolean
|     |  +--rw local-multiplier?                    multiplier
|     |  +--rw (interval-config-type)?
|     |     +--:(tx-rx-intervals)
|     |     |  +--rw desired-min-tx-interval?    uint32
|     |     |  +--rw required-min-rx-interval?   uint32
|     |     +--:(single-interval) {single-minimum-interval}?
|     |        +--rw min-interval?               uint32
|     +--rw afs {nlpid-control}?
|     |  +--rw af* [af]
|     |     +--rw af    iana-rt-types:address-family
|     +--rw mpls
|     |  +--rw ldp
|     |     +--rw igp-sync?   boolean {ldp-igp-sync}?
|     +--rw fast-reroute {fast-reroute}?
|     |  +--rw lfa {lfa}?
|     |     +--rw candidate-disabled?   boolean
|     |     +--rw enable?               boolean
|     |     +--rw remote-lfa {remote-lfa}?
|     |     |  +--rw enable?   boolean
|     |     +--rw level-1
|     |     |  +--rw candidate-disabled?   boolean
|     |     |  +--rw enable?               boolean
|     |     |  +--rw remote-lfa {remote-lfa}?
|     |     |     +--rw enable?   boolean
|     |     +--rw level-2
|     |        +--rw candidate-disabled?   boolean
|     |        +--rw enable?               boolean
|     |        +--rw remote-lfa {remote-lfa}?
|     |           +--rw enable?   boolean
```

```
|     +--rw topologies {multi-topology}?
|     |  +--rw topology* [name]
|     |     +--rw name
|     |           -> ../../../../../../../../rt:ribs/rib/name
|     |     +--rw metric
|     |        +--rw value?     wide-metric
|     |        +--rw level-1
|     |        |  +--rw value?   wide-metric
|     |        +--rw level-2
|     |           +--rw value?   wide-metric
|     +--ro adjacencies
|     |  +--ro adjacency* []
|     |     +--ro neighbor-sys-type?              level
|     |     +--ro neighbor-sysid?                 system-id
|     |     +--ro neighbor-extended-circuit-id?
|     |             extended-circuit-id
|     |     +--ro neighbor-snpa?                  snpa
|     |     +--ro usage?                          level
|     |     +--ro hold-timer?
|     |             rt-types:timer-value-seconds16
|     |     +--ro neighbor-priority?              uint8
|     |     +--ro lastuptime?                     yang:timestamp
|     |     +--ro state?                          adj-state-type
|     +--ro event-counters
|     |  +--ro adjacency-changes?            uint32
|     |  +--ro adjacency-number?             uint32
|     |  +--ro init-fails?                   uint32
|     |  +--ro adjacency-rejects?            uint32
|     |  +--ro id-len-mismatch?              uint32
|     |  +--ro max-area-addresses-mismatch?  uint32
|     |  +--ro authentication-type-fails?    uint32
|     |  +--ro authentication-fails?         uint32
|     |  +--ro lan-dis-changes?              uint32
|     +--ro packet-counters
|        +--ro level* [level]
|           +--ro level      level-number
|           +--ro iih
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
|           +--ro ish
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
|           +--ro esh
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
|           +--ro lsp
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
```

```
|           +--ro psnp
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
|           +--ro csnp
|           |  +--ro in?    uint32
|           |  +--ro out?   uint32
|           +--ro unknown
|              +--ro in?    uint32
|              +--ro out?   uint32
+--ro spf-log
|  +--ro event* [id]
|     +--ro id                     uint32
|     +--ro spf-type?              enumeration
|     +--ro level?                 level-number
|     +--ro schedule-timestamp?    yang:timestamp
|     +--ro start-timestamp?       yang:timestamp
|     +--ro end-timestamp?         yang:timestamp
|     +--ro trigger-lsp* [lsp]
|        +--ro lsp          lsp-id
|        +--ro sequence?    uint32
+--ro lsp-log
|  +--ro event* [id]
|     +--ro id                     uint32
|     +--ro level?                 level-number
|     +--ro lsp
|     |  +--ro lsp?         lsp-id
|     |  +--ro sequence?    uint32
|     +--ro received-timestamp?    yang:timestamp
|     +--ro change?                identityref
+--ro hostnames
|  +--ro hostname* [system-id]
|     +--ro system-id    system-id
|     +--ro hostname?    string
+--ro database
|  +--ro level-db* [level]
|     +--ro level    level-number
|     +--ro lsp* [lsp-id]
|        +--ro decoded-completed?              boolean
|        +--ro raw-data?                       yang:hex-string
|        +--ro lsp-id                          lsp-id
|        +--ro checksum?                       uint16
|        +--ro remaining-lifetime?             uint16
|        +--ro sequence?                       uint32
|        +--ro attributes?                     bits
|        +--ro ipv4-addresses*                 inet:ipv4-address
|        +--ro ipv6-addresses*                 inet:ipv6-address
|        +--ro ipv4-te-routerid?               inet:ipv4-address
|        +--ro ipv6-te-routerid?               inet:ipv6-address
```

```
|        +--ro protocol-supported*              uint8
|        +--ro dynamic-hostname?                string
|        +--ro authentication
|        |  +--ro authentication-type?   string
|        |  +--ro authentication-key?    string
|        +--ro mt-entries
|        |  +--ro topology* []
|        |     +--ro MT-ID?        uint16
|        |     +--ro attributes?   bits
|        +--ro router-capabilities* []
|        |  +--ro flags?       bits
|        |  +--ro node-tags {node-tag}?
|        |  |  +--ro node-tag* []
|        |  |     +--ro tag?   uint32
|        |  +--ro binary?      binary
|        +--ro is-neighbor
|        |  +--ro neighbor* []
|        |     +--ro neighbor-id?      system-id
|        |     +--ro i-e?              boolean
|        |     +--ro default-metric?   std-metric
|        |     +--ro delay-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro expense-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro error-metric
|        |        +--ro metric?      std-metric
|        |        +--ro supported?   boolean
|        +--ro extended-is-neighbor
|        |  +--ro neighbor* []
|        |     +--ro neighbor-id?   system-id
|        |     +--ro metric?        wide-metric
|        +--ro ipv4-internal-reachability
|        |  +--ro prefixes* []
|        |     +--ro up-down?          boolean
|        |     +--ro i-e?              boolean
|        |     +--ro ip-prefix?        inet:ipv4-address
|        |     +--ro prefix-len?       uint8
|        |     +--ro default-metric?   std-metric
|        |     +--ro delay-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro expense-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro error-metric
|        |        +--ro metric?      std-metric
```

```
|        |        +--ro supported?   boolean
|        +--ro ipv4-external-reachability
|        |  +--ro prefixes* []
|        |     +--ro up-down?          boolean
|        |     +--ro i-e?              boolean
|        |     +--ro ip-prefix?        inet:ipv4-address
|        |     +--ro prefix-len?       uint8
|        |     +--ro default-metric?   std-metric
|        |     +--ro delay-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro expense-metric
|        |     |  +--ro metric?      std-metric
|        |     |  +--ro supported?   boolean
|        |     +--ro error-metric
|        |        +--ro metric?      std-metric
|        |        +--ro supported?   boolean
|        +--ro extended-ipv4-reachability
|        |  +--ro prefixes* []
|        |     +--ro up-down?                boolean
|        |     +--ro ip-prefix?              inet:ipv4-address
|        |     +--ro prefix-len?             uint8
|        |     +--ro metric?                 wide-metric
|        |     +--ro tag*                    uint32
|        |     +--ro tag64*                  uint64
|        |     +--ro external-prefix-flag?   boolean
|        |     +--ro readvertisement-flag?   boolean
|        |     +--ro node-flag?              boolean
|        |     +--ro ipv4-source-router-id?  inet:ipv4-address
|        |     +--ro ipv6-source-router-id?  inet:ipv6-address
|        +--ro mt-is-neighbor
|        |  +--ro neighbor* []
|        |     +--ro mt-id?         uint16
|        |     +--ro neighbor-id?   system-id
|        |     +--ro metric?        wide-metric
|        +--ro mt-extended-ipv4-reachability
|        |  +--ro prefixes* []
|        |     +--ro mt-id?                  uint16
|        |     +--ro up-down?                boolean
|        |     +--ro ip-prefix?              inet:ipv4-address
|        |     +--ro prefix-len?             uint8
|        |     +--ro metric?                 wide-metric
|        |     +--ro tag*                    uint32
|        |     +--ro tag64*                  uint64
|        |     +--ro external-prefix-flag?   boolean
|        |     +--ro readvertisement-flag?   boolean
|        |     +--ro node-flag?              boolean
|        |     +--ro ipv4-source-router-id?  inet:ipv4-address
```

```
|        |     +--ro ipv6-source-router-id?   inet:ipv6-address
|        +--ro mt-ipv6-reachability
|        |  +--ro prefixes* []
|        |     +--ro MT-ID?                   uint16
|        |     +--ro up-down?                 boolean
|        |     +--ro ip-prefix?               inet:ipv6-address
|        |     +--ro prefix-len?              uint8
|        |     +--ro metric?                  wide-metric
|        |     +--ro tag*                     uint32
|        |     +--ro tag64*                   uint64
|        |     +--ro external-prefix-flag?    boolean
|        |     +--ro readvertisement-flag?    boolean
|        |     +--ro node-flag?               boolean
|        |     +--ro ipv4-source-router-id?   inet:ipv4-address
|        |     +--ro ipv6-source-router-id?   inet:ipv6-address
|        +--ro ipv6-reachability
|           +--ro prefixes* []
|              +--ro up-down?                 boolean
|              +--ro ip-prefix?              inet:ipv6-address
|              +--ro prefix-len?             uint8
|              +--ro metric?                 wide-metric
|              +--ro tag*                    uint32
|              +--ro tag64*                  uint64
|              +--ro external-prefix-flag?   boolean
|              +--ro readvertisement-flag?   boolean
|              +--ro node-flag?              boolean
|              +--ro ipv4-source-router-id?  inet:ipv4-address
|              +--ro ipv6-source-router-id?  inet:ipv6-address
+--ro local-rib
|  +--ro route* [prefix]
|     +--ro prefix       inet:ip-prefix
|     +--ro next-hops
|     |  +--ro next-hop* [next-hop]
|     |     +--ro outgoing-interface?   if:interface-ref
|     |     +--ro next-hop              inet:ip-address
|     +--ro metric?      uint32
|     +--ro level?       level-number
|     +--ro route-tag?   uint32
+--ro system-counters
   +--ro level* [level]
      +--ro level                              level-number
      +--ro corrupted-lsps?                    uint32
      +--ro authentication-type-fails?         uint32
      +--ro authentication-fails?              uint32
      +--ro database-overload?                 uint32
      +--ro own-lsp-purge?                     uint32
      +--ro manual-address-drop-from-area?     uint32
      +--ro max-sequence?                      uint32
```

```
            +--ro sequence-number-skipped?         uint32
            +--ro id-len-mismatch?                 uint32
            +--ro partition-changes?               uint32
            +--ro lsp-errors?                      uint32
            +--ro spf-runs?                        uint32

  rpcs:
    +---x clear-adjacency
    |  +---w input
    |     +---w routing-protocol-instance-name    instance-state-ref
    |     +---w level?                            level
    |     +---w interface?                        string
    +---x clear-database
       +---w input
          +---w routing-protocol-instance-name    instance-state-ref
          +---w level?                            level

  notifications:
    +---n database-overload
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro overload?                enumeration
    +---n lsp-too-large
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro pdu-size?                uint32
    |  +--ro lsp-id?                  lsp-id
    +---n if-state-change
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro state?                   if-state-type
    +---n corrupted-lsp-detected
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro lsp-id?                  lsp-id
    +---n attempt-to-exceed-max-sequence
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
```

```
    |  +--ro isis-level?              level
    |  +--ro lsp-id?                  lsp-id
    +---n id-len-mismatch
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro pdu-field-len?           uint8
    |  +--ro raw-pdu?                 binary
    +---n max-area-addresses-mismatch
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro max-area-addresses?      uint8
    |  +--ro raw-pdu?                 binary
    +---n own-lsp-purge
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro lsp-id?                  lsp-id
    +---n sequence-number-skipped
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro lsp-id?                  lsp-id
    +---n authentication-type-failure
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro raw-pdu?                 binary
    +---n authentication-failure
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
```

```
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro raw-pdu?                 binary
    +---n version-skew
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro protocol-version?        uint8
    |  +--ro raw-pdu?                 binary
    +---n area-mismatch
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro raw-pdu?                 binary
    +---n rejected-adjacency
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro raw-pdu?                 binary
    |  +--ro reason?                  string
    +---n protocols-supported-mismatch
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro raw-pdu?                 binary
    |  +--ro protocols*               uint8
    +---n lsp-error-detected
    |  +--ro routing-instance?        string
    |  +--ro routing-protocol-name?   string
    |  +--ro isis-level?              level
    |  +--ro interface-name?          string
    |  +--ro interface-level?         level
    |  +--ro extended-circuit-id?     extended-circuit-id
    |  +--ro lsp-id?                  lsp-id
    |  +--ro raw-pdu?                 binary
```

```
    |  +--ro error-offset?           uint32
    |  +--ro tlv-type?               uint8
    +---n adjacency-state-change
    |  +--ro routing-instance?       string
    |  +--ro routing-protocol-name?  string
    |  +--ro isis-level?             level
    |  +--ro interface-name?         string
    |  +--ro interface-level?        level
    |  +--ro extended-circuit-id?    extended-circuit-id
    |  +--ro neighbor?               string
    |  +--ro neighbor-system-id?     system-id
    |  +--ro state?                  adj-state-type
    |  +--ro reason?                 string
    +---n lsp-received
    |  +--ro routing-instance?       string
    |  +--ro routing-protocol-name?  string
    |  +--ro isis-level?             level
    |  +--ro interface-name?         string
    |  +--ro interface-level?        level
    |  +--ro extended-circuit-id?    extended-circuit-id
    |  +--ro lsp-id?                 lsp-id
    |  +--ro sequence?               uint32
    |  +--ro received-timestamp?     yang:timestamp
    |  +--ro neighbor-system-id?     system-id
    +---n lsp-generation
       +--ro routing-instance?       string
       +--ro routing-protocol-name?  string
       +--ro isis-level?             level
       +--ro lsp-id?                 lsp-id
       +--ro sequence?               uint32
       +--ro send-timestamp?         yang:timestamp
```

2.5. Authentication Parameters

The module enables authentication configuration through the IETF key-chain module ([I-D.ietf-rtgwg-yang-key-chain]). The IS-IS module imports the "ietf-key-chain" module and reuses some groupings to allow global and per interface configuration of authentication. If a global authentication is configured, an implementation SHOULD authenticate PSNPs, CSNPs and LSPs with the authentication parameters supplied. The authentication of hello PDUs can be activated on a per interface basis.

2.6. IGP/LDP synchronization

[RFC5443] defines a mechanism where IGP needs to be synchronized with LDP. An "ldp-igp-sync" feature has been defined in the model to support this mechanism. The "mpls/ldp/igp-sync" leaf under

"interface" allows activation of the mechanism on a per interface basis. The "mpls/ldp/igp-sync" container in the global configuration is empty on purpose and is not required for the activation. The goal of this empty container is to allow easy augmentation with additional parameters like timers for example.

2.7. ISO parameters

As IS-IS protocol is based on ISO protocol suite, some ISO parameters may be required.

This module augments interface configuration model to support ISO configuration parameters.

The clns-mtu can be defined under the interface.

2.8. IP FRR

This YANG model supports LFA ([RFC5286]) and remote LFA ([RFC7490]) as IP FRR techniques. The "fast-reroute" container may be augmented by other models to support other IPFRR flavors (MRT, TILFA ...).

The current version of the model supports activation of LFA and remote LFA at interface only. The global "lfa" container is present but kept empty to allow augmentation with vendor specific properties like policies.

Remote LFA is considered as a child of LFA. Remote LFA cannot be enabled if LFA is not enabled.

The "candidate-disabled" allows to mark an interface to not be used as a backup.

2.9. Operational States

Operational states are provided in the module in various places:

- system-counters: provides statistical informations about the global system.
- interface : provides configuration state informations for each interface.
- adjacencies: provides state informations about current IS-IS adjacencies.
- spf-log: provides informations about SPF events on the node. This SHOULD be implemented as a wrapping buffer.

- o lsp-log: provides informations about LSP events on the node (reception of an LSP or modification of local LSP). This SHOULD be implemented as a wrapping buffer and an implementation MAY decide to log refresh LSPs or not.

- o local-rib: provides the IS-IS internal routing table view.

- o database: provides details on the current LSDB.

- o hostnames: provides informations about system-id to hostname mappings.

- o fast-reroute: provides informations about IP FRR.

3. RPC Operations

The "ietf-isis" module defines two RPC operations:

- o clear-isis-database: reset the content of a particular IS-IS database and restart database synchronization with the neighbors.

- o clear-isis-adjacency: restart a particular set of IS-IS adjacencies.

4. Notifications

The "ietf-isis" module introduces some notifications :

database-overload : raised when overload condition is changed.

lsp-too-large : raised when the system tries to propagate a too large PDU.

corrupted-lsp-detected : raised when the system find that an LSP that was stored in memory has become corrupted.

attempt-to-exceed-max-sequence : This notification is sent when the system wraps the 32-bit sequence counter of an LSP.

id-len-mismatch : This notification is sent when we receive a PDU with a different value for the System ID length.

max-area-addresses-mismatch : This notification is sent when we receive a PDU with a different value for the Maximum Area Addresses.

own-lsp-purge : This notification is sent when the system receives a PDU with its own system ID and zero age.

sequence-number-skipped : This notification is sent when the system receives a PDU with its own system ID and different contents. The system has to reissue the LSP with a higher sequence number.

authentication-type-failure : This notification is sent when the system receives a PDU with the wrong authentication type field.

authentication-failure : This notification is sent when the system receives a PDU with the wrong authentication information.

version-skew : This notification is sent when the system receives a PDU with a different protocol version number.

area-mismatch : This notification is sent when the system receives a Hello PDU from an IS that does not share any area address.

rejected-adjacency : This notification is sent when the system receives a Hello PDU from an IS but does not establish an adjacency for some reason.

protocols-supported-mismatch : This notification is sent when the system receives a non pseudonode LSP that has no matching protocol supported.

lsp-error-detected : This notification is sent when the system receives a LSP with a parse error.

adjacency-change : This notification is sent when an IS-IS adjacency moves to Up state or to Down state.

lsp-received : This notification is sent when a LSP is received.

lsp-generation : This notification is sent when a LSP is regenerated.

5. Interaction with Other YANG Modules

The "isis" configuration container augments the "/rt:routing/ rt:control-plane-protocols/control-plane-protocol" container of the ietf-routing [I-D.ietf-netmod-routing-cfg] module by defining IS-IS specific parameters.

The "isis" module augments "/if:interfaces/if:interface" with ISO specific parameters.

The "isis" operational state container augments the "/rt:routing-state/rt:control-plane-protocols/control-plane-protocol" container of the ietf-routing module by defining IS-IS specific operational states.

Some IS-IS specific routes attributes are added to route objects of the ietf-routing module by augmenting "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route".

The modules defined in this document use some groupings from ietf-keychain [I-D.ietf-rtgwg-yang-key-chain].

6. IS-IS YANG Module

<CODE BEGINS> file "ietf-isis@2018-08-09.yang"

```
module ietf-isis {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-isis";

  prefix isis;

  import ietf-routing {
    prefix "rt";
    reference "RFC 8349 - A YANG Data Model for Routing
               Management (NMDA Version)";
  }

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6021 - Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6021 - Common YANG Data Types";
  }

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343 - A YANG Data Model for Interface
               Management (NDMA Version)";
  }

  import ietf-key-chain {
```

```
      prefix "key-chain";
      reference "RFC 8177 - YANG Data Model for Key Chains";
    }

    import ietf-routing-types {
      prefix "rt-types";
      reference "RFC 8291 - Common YANG Data Types for the
                 Routing Area";
    }

    import iana-routing-types {
      prefix "iana-rt-types";
      reference "RFC 8291 - Common YANG Data Types for the
                 Routing Area";
    }

    import ietf-bfd-types {
      prefix "bfd-types";
      reference "RFC XXXX - YANG Data Model for Bidirectional
                 Forwarding Detection (BFD)";
    }

    organization
      "IETF IS-IS Working Group";

    contact
      "WG List:  &lt;mailto:isis-wg@ietf.org&gt;

      Editor:    Stephane Litkowski
            &lt;mailto:stephane.litkowski@orange.com&gt;

          Derek Yeung
            &lt;mailto:derek@arrcus.com&gt;
          Acee Lindem
            &lt;mailto:acee@cisco.com&gt;
          Jeffrey Zhang
            &lt;mailto:zzhang@juniper.net&gt;
          Ladislav Lhotka
            &lt;mailto:llhotka@nic.cz&gt;
          Yi Yang
            &lt;mailto:yiya@cisco.com&gt;
          Dean Bogdanovic
            &lt;mailto:deanb@juniper.net&gt;
          Kiran Agrahara Sreenivasa
            &lt;mailto:kkoushik@brocade.com&gt;
          Yingzhen Qu
            &lt;mailto:yiqu@cisco.com&gt;
                 Jeff Tantsura
```

```
                  &lt;mailto:jefftant.ietf@gmail.com&gt;

      ";

    description
      "The YANG module defines a generic configuration model for
       IS-IS common across all of the vendor implementations.";

    revision 2018-08-09 {
      description
        "Initial revision.";
      reference "RFC XXXX";
    }

    /* Identities */

    identity isis {
      base rt:routing-protocol;
      description "Identity for the IS-IS routing protocol.";
    }

    identity isis-adjacency-change {
      description "Identity for the IS-IS routing protocol
                   adjacency state.";
    }

    identity clear-isis-database {
      description "Identity for the IS-IS routing protocol
                   database reset action.";
    }

    identity clear-isis-adjacency {
      description "Identity for the IS-IS routing protocol
                   adjacency reset action.";
    }

    identity lsp-log-reason {
      description "Base identity for an LSP change log reason.";
    }

    identity refresh {
      base lsp-log-reason;
      description
        "Identity used when the LSP log reason is
         a refresh LSP received.";
    }

    identity content-change {
```

```
      base lsp-log-reason;
      description
        "Identity used when the LSP log reason is
         a change in the content of the LSP.";
    }


    /* Feature definitions */

    feature ietf-spf-delay {
      description
        "Support of IETF SPF delay algorithm.";
      reference "RFC XXXX - SPF Back-off algorithm for link
                 state IGPs";
    }
    feature bfd {
      description
        "Support for BFD detection of IS-IS neighbor reachability.";
      reference "RFC 5880 - Bidirectional Forwarding Detection (BFD)
                 RFC 5881 - Bidirectional Forwarding Detection
                 (BFD) for IPv4 and IPv6 (Single Hop)";
    }
    feature key-chain {
      description
        "Support of keychain for authentication.";
      reference "RFC8177 - YANG Data Model for Key Chains";
    }
    feature node-flag {
      description
        "Support of node-flag advertisement
         as prefix attribute";
      reference "RFC7794 - IS-IS Prefix Attributes for
                 Extended IP and IPv6 Reachability";
    }
    feature node-tag {
      description
        "Support of node tag.";
      reference "RFC7917 - Advertising Node Administrative Tags
                 in IS-IS";
    }
    feature ldp-igp-sync {
      description
        "Support of synchronization between IS-IS and LDP.";
      reference "RFC5443 - LDP IGP Synchronization";
    }
    feature fast-reroute {
      description
        "Support of IP Fast Reroute.";
```

```
    }
    feature nsr {
      description
        "Support of Non-Stop Routing.";
    }
    feature lfa {
      description
        "Support of Loop-Free Alternates.";
      reference "RFC5286 - Basic Specification of IP Fast-Reroute:
                 Loop-free Alternates";
    }
    feature remote-lfa {
      description
        "Support of remote Loop Free Alternates.";
      reference "RFC7490 - Remote Loop-Free Alternate Fast Reroute";
    }

    feature overload-max-metric {
      description
        "Support of overload by setting
         all links to max metric.";
    }
    feature prefix-tag {
      description
        "Support for 32-bit prefix tags";
      reference "RFC5130 - A Policy Control Mechanism in
                 IS-IS Using Administrative Tags";
    }
    feature prefix-tag64 {
      description
        "Support for 64-bit prefix tags";
      reference "RFC5130 - A Policy Control Mechanism in
                 IS-IS Using Administrative Tags";
    }
    feature auto-cost {
      description
        "Use an automated assignment of metrics.";
    }

    feature te-rid {
      description
        "Traffic-Engineering Router-ID.";
      reference "RFC5305 - IS-IS Extensions for Traffic Engineering
                 RFC6119 - IPv6 Traffic Engineering in IS-IS";
    }
    feature max-ecmp {
      description
        "Setting maximum number of ECMP paths.";
```

```
  }
  feature multi-topology {
    description
      "Multitopology routing support.";
    reference "RFC5120 - M-IS-IS: Multi Topology Routing in IS-IS";
  }
  feature nlpid-control {
    description
      "This feature controls the advertisement
       of support NLPID within IS-IS configuration.";
  }
  feature graceful-restart {
    description
      "Graceful restart support.";
    reference "RFC5306 - Restart Signaling in IS-IS";
  }

  feature lsp-refresh {
    description
      "Configuration of LSP refresh interval.";
  }

  feature maximum-area-addresses {
    description
      "Support of maximum-area-addresses config.";
  }

  feature admin-control {
    description
      "Control administrative state of IS-IS.";
  }

  /* Type definitions */


  typedef instance-state-ref {
    type leafref {
      path "/rt:routing-state/"
        +"rt:control-plane-protocols/rt:control-plane-protocol/"
        +"rt:name";
    }
    description
      "This type is used for leaves that reference state data for
       an IS-IS protocol instance.";
  }

  typedef circuit-id {
    type uint8;
```

```
      description
        "This type defines the circuit ID
         associated with an interface.";
    }

    typedef extended-circuit-id {
      type uint32;
      description
        "This type defines the extended circuit ID
         associated with an interface.";
    }

    typedef interface-type {
      type enumeration {
        enum broadcast {
          description
            "Broadcast interface type.";
        }
        enum point-to-point {
          description
            "Point-to-point interface type.";
        }
      }
      description
        "This type defines the type of adjacency
         to be established on the interface.
         The interface-type determines the type
         of hello message that is used.";

    }

    typedef level {
      type enumeration {
        enum "level-1" {
          description
            "This enum indicates L1-only capability.";
        }
        enum "level-2" {
          description
            "This enum indicates L2-only capability.";
        }
        enum "level-all" {
          description
            "This enum indicates capability for both levels.";
        }
      }
      default "level-all";
      description
```

```
        "This type defines IS-IS level of an object.";

    }

    typedef adj-state-type {
      type enumeration {
        enum "up" {
          description
            "State indicates the adjacency is established.";
        }
        enum "down" {
          description
            "State indicates the adjacency is NOT established.";
        }
        enum "init" {
          description
            "State indicates the adjacency is establishing.";
        }
        enum "failed" {
          description
            "State indicates the adjacency is failed.";
        }
      }
      description
        "This type defines states of an adjacency";
    }

    typedef if-state-type {
      type enumeration {
        enum "up" {
          description "Up state.";

        }
        enum "down" {
          description "Down state";
        }
      }
      description
        "This type defines the state of an interface";
    }

    typedef level-number {
      type uint8 {
        range "1 .. 2";
      }
      description
        "This type defines the current IS-IS level.";
    }
```

```
     typedef lsp-id {
       type string {
         pattern
           '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]'
          +'{4}\.[0-9][0-9]-[0-9][0-9]';
       }
       description
         "This type defines the IS-IS LSP ID format using a
          pattern, An example LSP ID is 0143.0438.AeF0.02-01";
     }

     typedef area-address {
       type string {
         pattern '[0-9A-Fa-f]{2}\.([0-9A-Fa-f]{4}\.){0,3}';
       }
       description
         "This type defines the area address format.";
     }

     typedef snpa {
       type string {
         length "0 .. 20";
       }
       description
         "This type defines the Subnetwork Point
          of Attachement (SNPA) format.";
     }

     typedef system-id {
       type string {
         pattern
           '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}';
       }
       description
         "This type defines IS-IS system-id using pattern,
          An example system-id is 0143.0438.AeF0";
     }

     typedef wide-metric {
       type uint32 {
         range "0 .. 16777215";
       }
       description
         "This type defines wide style format of IS-IS metric.";
     }

     typedef std-metric {
       type uint8 {
```

```
        range "0 .. 63";
      }
      description
        "This type defines old style format of IS-IS metric.";
    }

    typedef mesh-group-state {
      type enumeration {
        enum "mesh-inactive" {
          description
            "Interface is not part of a mesh group.";
        }
        enum "mesh-set" {
          description
            "Interface is part of a mesh group.";
        }
        enum "mesh-blocked" {
          description
            "LSPs must not be flooded over this interface.";
        }
      }
      description
        "This type describes mesh group state of an interface";
    }


    /* Grouping definitions for configuration and ops state */


    grouping adjacency-state {
      container adjacencies {
        config false;
        list adjacency {
          leaf neighbor-sys-type {
            type level;
            description
              "Level capability of neighboring system";
          }
          leaf neighbor-sysid {
            type system-id;
            description
              "The system-id of the neighbor";
          }
          leaf neighbor-extended-circuit-id {
            type extended-circuit-id;
            description
              "Circuit ID of the neighbor";
          }
```

```
            leaf neighbor-snpa {
              type snpa;
              description
                "SNPA of the neighbor";
            }
            leaf usage {
              type level;
              description
                "Define the level(s) activated on the adjacency.
                 On a p2p link this might be level 1 and 2,
                 but on a LAN, the usage will be level 1
                 between peers at level 1 or level 2 between
                 peers at level 2.";
            }
            leaf hold-timer {
              type rt-types:timer-value-seconds16;
              units seconds;
              description
                "The holding time in seconds for this
                 adjacency. This value is based on
                 received hello PDUs and the elapsed
                 time since receipt.";
            }
            leaf neighbor-priority {
              type uint8 {
                range "0 .. 127";
              }
              description
                "Priority of the neighboring IS for becoming
                 the DIS.";
            }
            leaf lastuptime {
              type yang:timestamp;
              description
                "When the adjacency most recently entered
                 state 'up', measured in hundredths of a
                 second since the last reinitialization of
                 the network management subsystem.
                 The value is 0 if the adjacency has never
                 been in state 'up'.";
            }
            leaf state {
              type adj-state-type;
              description
                "This leaf describes the state of the interface.";
            }

            description
```

```
            "List of operational adjacencies.";
        }
        description
          "This container lists the adjacencies of
           the local node.";
      }
      description
        "Adjacency state";
    }

    grouping fast-reroute-global-state {
      container protected-routes {
        config false;
        list af-stats {
          key "af prefix alternate";

          leaf af {
            type iana-rt-types:address-family;
            description
              "Address-family";
          }
          leaf prefix {
            type string;
            description
              "Protected prefix.";
          }
          leaf alternate {
            type string;
            description
              "Alternate nexthop for the prefix.";
          }
          leaf alternate-type {
            type enumeration {
              enum equal-cost {
                description
                  "ECMP alternate.";
              }
              enum lfa {
                description
                  "LFA alternate.";
              }
              enum remote-lfa {
                description
                  "Remote LFA alternate.";
              }
              enum tunnel {
                description
                  "Tunnel based alternate
```

```
                (like RSVP-TE or GRE).";
            }
            enum ti-lfa {
              description
                "TI-LFA alternate.";
            }
            enum mrt {
              description
                "MRT alternate.";
            }
            enum other {
              description
                "Unknown alternate type.";
            }
          }
          description
            "Type of alternate.";
        }
        leaf best {
          type boolean;
          description
            "Indicates if the alternate is the preferred.";
        }
        leaf non-best-reason {
          type string;
          description
            "Information field to describe why the alternate
             is not best.";
        }
        leaf protection-available {
          type bits {
            bit node-protect {
              position 0;
              description
                "Node protection available.";
            }
            bit link-protect {
              position 1;
              description
                "Link protection available.";
            }
            bit srlg-protect {
              position 2;
              description
                "SRLG protection available.";
            }
            bit downstream-protect {
              position 3;
```

```
                description
                  "Downstream protection available.";
              }
              bit other {
                position 4;
                description
                  "Other protection available.";
              }
            }
            description "Protection provided by the alternate.";
          }
          leaf alternate-metric1 {
            type uint32;
            description
              "Metric from Point of Local Repair (PLR) to
               destination through the alternate path.";
          }
          leaf alternate-metric2 {
            type uint32;
            description
              "Metric from PLR to the alternate node";
          }
          leaf alternate-metric3 {
            type uint32;
            description
              "Metric from alternate node to the destination";
          }
          description
            "Per-AF protected prefix statistics.";
        }
        description
          "List of prefixes that are protected.";
      }

      container unprotected-routes {
        config false;
        list af-stats {
          key "af prefix";

          leaf af {
            type iana-rt-types:address-family;

            description "Address-family";
          }
          leaf prefix {
            type string;
            description "Unprotected prefix.";
          }
```

```
            description
              "Per AF unprotected prefix statistics.";
          }
          description
            "List of prefixes that are not protected.";
        }

        list protection-statistics {
          key frr-protection-method;
          config false;
          leaf frr-protection-method {
            type string;
            description "Protection method used.";
          }
          list af-stats {
            key af;

            leaf af {
              type iana-rt-types:address-family;

              description "Address-family";
            }
            leaf total-routes {
              type uint32;
              description "Total prefixes.";
            }
            leaf unprotected-routes {
              type uint32;
              description
                "Total prefixes that are not protected.";
            }
            leaf protected-routes {
              type uint32;
              description
                "Total prefixes that are protected.";
            }
            leaf linkprotected-routes {
              type uint32;
              description
                "Total prefixes that are link protected.";
            }
            leaf nodeprotected-routes {
              type uint32;
              description
                "Total prefixes that are node protected.";
            }
            description
              "Per AF protected prefix statistics.";
```

```
        }

        description "Global protection statistics.";
      }
      description "IPFRR states.";
    }

    grouping notification-instance-hdr {
      description
        "Instance specific IS-IS notification data grouping";
      leaf routing-instance {
        type string;
        description "Name of the routing-instance instance.";
      }
      leaf routing-protocol-name {
        type string;
        description "Name of the IS-IS instance.";
      }
      leaf isis-level {
        type level;
        description "IS-IS level of the instance.";
      }
    }

    grouping notification-interface-hdr {
      description
        "Interface specific IS-IS notification data grouping";
      leaf interface-name {
        type string;
        description "IS-IS interface name";
      }
      leaf interface-level {
        type level;
        description "IS-IS level of the interface.";
      }
      leaf extended-circuit-id {
        type extended-circuit-id;
        description "Eextended circuit-id of the interface.";
      }
    }

    grouping route-content {
      description
        "IS-IS protocol-specific route properties grouping.";
      leaf metric {
        type uint32;
        description "IS-IS metric of a route.";
      }
```

```
        leaf-list tag {
          type uint64;
          description
            "List of tags associated with the route. The leaf
             describes both 32-bit and 64-bit tags.";
        }
        leaf route-type {
          type enumeration {
            enum l2-up-internal {
              description "Level 2 internal route
                           and not leaked to a lower level";
            }
            enum l1-up-internal {
              description "Level 1 internal route
                           and not leaked to a lower level";
            }
            enum l2-up-external {
              description "Level 2 external route
                           and not leaked to a lower level";
            }
            enum l1-up-external {
              description "Level 1 external route
                           and not leaked to a lower level";
            }
            enum l2-down-internal {
              description "Level 2 internal route
                           and leaked to a lower level";
            }
            enum l1-down-internal {
              description "Level 1 internal route
                           and leaked to a lower level";
            }
            enum l2-down-external {
              description "Level 2 external route
                           and leaked to a lower level";
            }
            enum l1-down-external {
              description "Level 1 external route
                           and leaked to a lower level";
            }
          }
          description "IS-IS route type.";
        }
      }

      grouping admin-control {
        leaf enable {
          if-feature admin-control;
```

```
        type boolean;
        default true;
        description
          "Control the administrative state.";
      }
      description
        "Grouping for admin control.";
    }

    grouping fast-reroute-global-cfg {
      description
        "IPFRR global configuration grouping";
      container lfa {
        if-feature lfa;
        description
          "This container may be augmented with global
           parameters for LFA. Creating the container has no
           effect on LFA activation.";
      }
    }

    grouping fast-reroute-if-cfg {
      description
        "IPFRR interface configuration grouping";
      container lfa {
        if-feature lfa;
        uses lfa-if-cfg;
        container level-1 {
          uses lfa-if-cfg;
          description
            "LFA level 1 config";
        }
        container level-2 {
          uses lfa-if-cfg;
          description
            "LFA level 2 config";
        }
        description
          "LFA config";
      }
    }

    grouping ietf-spf-delay-cfg {
      leaf initial-delay {
        type rt-types:timer-value-milliseconds;
        units msec;
        description
          "Delay used while in QUIET state.";
```

```
      }
      leaf short-delay {
        type rt-types:timer-value-milliseconds;
        units msec;
        description
          "Delay used while in SHORT_WAIT state.";
      }
      leaf long-delay {
        type rt-types:timer-value-milliseconds;
        units msec;
        description
          "Delay used while in LONG_WAIT state.";
      }

      leaf hold-down {
        type rt-types:timer-value-milliseconds;
        units msec;
        description
          "Timer used to consider an IGP stability period.";
      }
      leaf time-to-learn {
        type rt-types:timer-value-milliseconds;
        units msec;
        description
          "Duration used to learn all the IGP events
           related to a single component failure.";
      }

      description
        "Grouping for IETF SPF delay configuration.";
    }

    grouping ietf-spf-delay-state {
      leaf current-state {
        type enumeration {
          enum "quiet" {
            description "QUIET state";
          }
          enum "short-wait" {
            description "SHORT_WAIT state";
          }
          enum "long-wait" {
            description "LONG_WAIT state";
          }
        }
        config false;
        description
          "Current SPF backoff algorithm state.";
```

```
      }
      leaf remaining-time-to-learn {
        type rt-types:timer-value-milliseconds;
        units "msec";
        config false;
        description
          "Remaining time until time-to-learn timer fires.";
      }
      leaf remaining-hold-down {
        type rt-types:timer-value-milliseconds;
        units "msec";
        config false;
        description
          "Remaining time until hold-down timer fires.";
      }
      leaf last-event-received {
        type yang:timestamp;
        config false;
        description
          "Time of last IGP event received";
      }
      leaf next-spf-time {
        type yang:timestamp;
        config false;
        description
          "Time when next SPF has been scheduled.";
      }
      leaf last-spf-time {
        type yang:timestamp;
        config false;
        description
          "Time of last SPF computation.";
      }
      description
        "Grouping for IETF SPF delay operational states.";
    }

    grouping local-rib {
      description "Local-rib grouping.";
      container local-rib {
        config false;
        description "Local-rib.";
        list route {
          key "prefix";
          description "List of IS-IS local RIB Routes";
          leaf prefix {
            type inet:ip-prefix;
            description "Destination prefix.";
```

```
          }
          container next-hops {
            description "All next hops for the route.";
            list next-hop {
              key "next-hop";
              description "List of next hop for the route";
              leaf outgoing-interface {
                type if:interface-ref;
                description
                  "Name of the outgoing interface.";
              }
              leaf next-hop {
                type inet:ip-address;
                description "Nexthop address.";
              }
            }
          }
          leaf metric {
            type uint32;
            description "Metric for this route.";
          }
          leaf level {
            type level-number;
            description "Level number for this route.";
          }
          leaf route-tag {
            type uint32;
            description "Route tag for this route.";
          }
        }
      }
    }

    grouping isis-node-tag-cfg {
      description "IS-IS node tag config.";
      container node-tags {
        if-feature node-tag;
        list node-tag {
          key tag;
          leaf tag {
            type uint32;
            description
              "Node tag value.";
          }
          description
            "List of tags.";
        }
        description
```

```
            "Container for node tags.";
        }
      }

      grouping authentication-global-cfg {
        choice authentication-type {
          case key-chain {
            if-feature key-chain;
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Reference to a key-chain.";
            }
          }
          case password {
            leaf key {
              type string;
              description
                "This leaf specifies the authentication key.";
            }
            leaf crypto-algorithm {
              type identityref {
                base key-chain:crypto-algorithm;
              }
              description
                "Cryptographic algorithm associated with key.";
            }
          }
          description "Choice of authentication.";
        }
        description "Grouping for global authentication config.";
      }

      grouping metric-type-global-cfg {
        leaf value {
          type enumeration {
            enum wide-only {
              description
                "Advertise new metric style only (RFC5305)";
            }
            enum old-only {
              description
                "Advertise old metric style only (RFC1195)";
            }
            enum both {
              description "Advertise both metricstyles";
            }
          }
```

```
        default wide-only;
        description
          "Type of metric to be generated:
           - wide-only means only new metric style
              is generated,
           - old-only means that only old style metric
             is generated,
           - both means that both are advertised.
           This leaf is only affecting IPv4 metrics.";
      }
      description
        "Grouping for global metric style config.";
    }

    grouping default-metric-global-cfg {
      leaf value {
        type wide-metric;
        default "10";
        description  "Value of the metric";
      }
      description
        "Global default metric config grouping.";
    }


    grouping overload-global-cfg {
      leaf status {
        type boolean;
        default false;
        description
          "This leaf specifies the overload status.";
      }
      description "Grouping for overload bit config.";
    }

    grouping overload-max-metric-global-cfg {
      leaf timeout {
        type rt-types:timer-value-seconds16;
        units "seconds";
        description
          "Timeout (in seconds) of the overload condition.";
      }
      description
        "Overload maximum metric configuration grouping";
    }

    grouping route-preference-global-cfg {
      choice granularity {
```

```
        case detail {
          leaf internal {
            type uint8;
            description
              "Protocol preference for internal routes.";
          }
          leaf external {
            type uint8;
            description
              "Protocol preference for external routes.";
          }
        }
        case coarse {
          leaf default {
            type uint8;
            description
              "Protocol preference for all IS-IS routes.";
          }
        }
        description
          "Choice for implementation of route preference.";
      }
      description
        "Global route preference grouping";
    }

    grouping hello-authentication-cfg {
      choice authentication-type {
        case key-chain {
          if-feature key-chain;
          leaf key-chain {
            type key-chain:key-chain-ref;
            description "Reference to a key-chain.";
          }
        }
        case password {
          leaf key {
            type string;
            description "Authentication key specification";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Cryptographic algorithm associated with key.";
          }
        }
```

```
        description "Choice of authentication.";
      }
      description "Grouping for hello authentication.";
    }

    grouping hello-interval-cfg {
      leaf value {
        type rt-types:timer-value-seconds16;
        units "seconds";
        default 10;
        description
          "Interval (in seconds) between successive hello
           messages.";
      }

      description "Interval between hello messages.";
    }

    grouping hello-multiplier-cfg {
      leaf value {
        type uint16;
        default 3;
        description
          "Number of missed hello messages prior to
           declaring the adjacency down.";
      }
      description
          "Number of missed hello messages prior to
           adjacency down grouping.";
    }

    grouping priority-cfg {
      leaf value {
        type uint8 {
          range "0 .. 127";
        }
        default 64;
        description
          "Priority of interface for DIS election.";
      }

      description "Interface DIS election priority grouping";
    }

    grouping metric-cfg {
      leaf value {
        type wide-metric;
        default "10";
```

```
        description "Metric value.";
      }
      description "Interface metric grouping";
    }

    grouping lfa-if-cfg {
      leaf candidate-disabled {
        type boolean;
        default false;
        description
          "Prevent the interface to be used as backup.";
      }
      leaf enable {
        type boolean;
        default false;
        description
          "LFA Activation - this model assumes activation
           of per-prefix LFA.";
      }

      container remote-lfa {
        if-feature remote-lfa;
        leaf enable {
          type boolean;
          default false;
          description
            "Activates rLFA.";
        }
        description "Remote LFA configuration.";
      }
      description "Grouping for LFA interface configuration";
    }

    grouping isis-global-cfg {
      description "IS-IS global configuration grouping";

      uses admin-control;

      leaf level-type {
        type level;
        default "level-all";
        description
          "Level of an IS-IS node - can be level-1-only,
           level-2-only or level-1-2.";
      }

      leaf system-id {
        type system-id;
```

```
        description "System-id of the node.";
      }

      leaf maximum-area-addresses {
        if-feature maximum-area-addresses;
        type uint8;
        default 3;
        description "Maximum areas supported.";
      }

      leaf-list area-address {
        type area-address;
        description
          "List of areas supported by the protocol instance.";
      }

      container mpls {
        container te-rid {
          if-feature te-rid;
          description
            "Stable ISIS Router IP Address used for Traffic
             Engineering";
          leaf ipv4-router-id {
            type inet:ipv4-address;
            description
              "Router ID value that would be used in TLV 134.";
          }
          leaf ipv6-router-id {
            type inet:ipv6-address;
            description
              "Router ID value that would be used in TLV 140.";
          }
        }
        container ldp {
          container igp-sync {
            if-feature ldp-igp-sync;
            description
              "This container may be augmented with global
               parameters for igp-ldp-sync.";
          }
          description "LDP configuration.";
        }
        description "MPLS configuration";
      }
      container auto-cost {
        if-feature auto-cost;
        leaf reference-bandwidth {
          type uint32;
```

```
            units "bps";
            description "Bandwidth for calculating metric.";
          }
          leaf enable {
            type boolean;
            default false;
            description "Enable/disable auto-cost.";
          }
          description "Auto-cost configuration.";
        }
        leaf lsp-mtu {
          type uint16;
          units "bytes";
          default 1492;
          description
            "Maximum size of an LSP PDU in bytes.";
        }
        leaf lsp-lifetime {
          type uint16 {
            range "1..65535";
          }
          units "seconds";
          description
            "Lifetime of the router's LSPs in seconds.";
        }
        leaf lsp-refresh {
          if-feature lsp-refresh;
          type rt-types:timer-value-seconds16;
          units "seconds";
          description
            "Refresh interval of the router's LSPs in seconds.";
        }
        container graceful-restart {
          if-feature graceful-restart;
          leaf enable {
            type boolean;
            default false;
            description "Enable graceful restart.";
          }
          leaf restart-interval {
            type rt-types:timer-value-seconds16;
            units "seconds";
            description
              "Interval (in seconds) to attempt graceful restart prior
               to failure.";
          }
          leaf helper-enable {
            type boolean;
```

```
            default true;
            description
              "Enable local IS-IS router as graceful restart helper.";
          }
          description "Graceful-Restart Configuration.";
        }

        container nsr {
          if-feature nsr;
          description "Non-Stop Routing (NSR) configuration.";
          leaf enable {
            type boolean;
            default false;
            description "Enable/Disable Non-Stop Routing (NSR).";
          }
        }

        uses isis-node-tag-cfg;

        container authentication {
          uses authentication-global-cfg;

          container level-1 {
            uses authentication-global-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses authentication-global-cfg;
            description "level-2 specific configuration";
          }
          description "Authentication global configuration for
                       both LSPs and SNPs.";
        }

        container metric-type {
          uses metric-type-global-cfg;
          container level-1 {
            uses metric-type-global-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses metric-type-global-cfg;
            description "level-2 specific configuration";
          }
          description "Metric style global configuration";
        }

        container default-metric {
```

```
          uses default-metric-global-cfg;
          container level-1 {
            uses default-metric-global-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses default-metric-global-cfg;
            description "level-2 specific configuration";
          }
          description "Default metric global configuration";
        }

        container afs {
          if-feature nlpid-control;
          list af {
            key af;
            leaf af {
              type iana-rt-types:address-family;
              description "Address-family";
            }
            leaf enable {
              type boolean;
              description "Activate the address family.";
            }
            description
              "List of address families and whether or not they
               are activated.";
          }
          description "Address Family configuration";
        }

        container preference {
          uses route-preference-global-cfg;
          description "Router preference configuration for IS-IS
                       protocol instance route installation";
        }

        container overload {
          uses overload-global-cfg;
          description "Router protocol instance overload state
                       configuration";
        }

        container overload-max-metric {
          if-feature overload-max-metric;
          uses overload-max-metric-global-cfg;
          description
            "Router protocol instance overload maximum
```

```
            metric advertisement configuration.";
        }
      }

      grouping isis-global-topologies-cfg {
        description "Per-topology configuration";
        container default-metric {
          uses default-metric-global-cfg;
          container level-1 {
            uses default-metric-global-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses default-metric-global-cfg;
            description "level-2 specific configuration";
          }
          description "Default metric per-topology configuration";
        }
        uses isis-node-tag-cfg;
      }

      grouping isis-if-cfg {
        description "Interface configuration grouping";
        leaf level-type {
          type level;
          default "level-all";
          description "IS-IS level of the interface.";
        }
        leaf lsp-pacing-interval {
          type rt-types:timer-value-milliseconds;
          units "milliseconds";
          default 33;
          description
            "Interval (in milli-seconds) between LSP
             transmissions.";
        }
        leaf lsp-retransmit-interval {
          type rt-types:timer-value-seconds16;
          units "seconds";
          description
            "Interval (in seconds) between LSP
             retransmissions.";
        }
        leaf passive {
          type boolean;
          default "false";
          description
            "Indicates whetherthe interface is in passive mode (IS-IS
```

```
           not running but network is advertised).";
        }
        leaf csnp-interval {
          type rt-types:timer-value-seconds16;
          units "seconds";
          default 10;
          description
             "Interval (in seconds) between CSNP messages.";
        }
        container hello-padding {
          leaf enable {
            type boolean;
            default "true";
            description
              "IS-IS Hello-padding activation - enabled by default.";
          }
          description "IS-IS hello padding configuration.";
        }
        leaf mesh-group-enable {
          type mesh-group-state;
          description "IS-IS interface mesh-group state";
        }
        leaf mesh-group {
          when "../mesh-group-enable = 'mesh-set'" {
            description
              "Only valid when mesh-group-enable equals meshset";
          }
          type uint8;
          description "IS-IS interface mesh-group ID.";
        }
        leaf interface-type {
          type interface-type;
          default "broadcast";
          description
            "Type of adjacency to be established on the interface. This
             dictates the type of hello messages that are used.";
        }

        uses admin-control;

        leaf-list tag {
          if-feature prefix-tag;
          type uint32;
          description
            "List of tags associated with the interface.";
        }
        leaf-list tag64 {
          if-feature prefix-tag64;
```

```
          type uint64;
          description
            "List of 64-bit tags associated with the interface.";
        }
        leaf node-flag {
          if-feature node-flag;
          type boolean;
          default false;
          description
            "Set prefix as a node representative prefix.";
        }
        container hello-authentication {
          uses hello-authentication-cfg;
          container level-1 {
            uses hello-authentication-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses hello-authentication-cfg;
            description "level-2 specific configuration";
          }
          description
            "Authentication type to be used in hello messages.";
        }
        container hello-interval {
          uses hello-interval-cfg;
          container level-1 {
            uses hello-interval-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses hello-interval-cfg;
            description "level-2 specific configuration";
          }
          description "Interval between hello messages.";
        }
        container hello-multiplier {
          uses hello-multiplier-cfg;
          container level-1 {
            uses hello-multiplier-cfg;
            description "level-1 specific configuration";
          }
          container level-2 {
            uses hello-multiplier-cfg;
            description "level-2 specific configuration";
          }
          description "Hello multiplier configuration.";
        }
```

```
      container priority {
        must '../interface-type = "broadcast"' {
          error-message
            "Priority only applies to broadcast interfaces.";
          description "Check for broadcast interface.";
        }
        uses priority-cfg;
        container level-1 {
          uses priority-cfg;
          description "level-1 specific configuration";
        }
        container level-2 {
          uses priority-cfg;
          description "level-2 specific configuration";
        }
        description "Priority for DIS election.";
      }
      container metric {
        uses metric-cfg;
        container level-1 {
          uses metric-cfg;
          description "level-1 specific configuration";
        }
        container level-2 {
          uses metric-cfg;
          description "level-2 specific configuration";
        }
        description "Metric configuration.";
      }
      container bfd {
        if-feature bfd;
        description "BFD Client Configuration.";
        uses bfd-types:client-cfg-parms;

        reference "draft-ietf-bfd-yang-xx.txt:
                   YANG Data Model for Bidirectional Forwarding
                   Detection (BFD)";
      }
      container afs {
        if-feature nlpid-control;
        list af {
          key af;
          leaf af {
            type iana-rt-types:address-family;
            description  "Address-family";
          }
          description "List of AFs.";
        }
```

```
        description "Interface address-families";
      }
      container mpls {
        container ldp {
          leaf igp-sync {
            if-feature ldp-igp-sync;
            type boolean;
            default false;
            description "Enables IGP/LDP synchronization";
          }
          description "LDP protocol related configuration.";
        }
        description "MPLS configuration for IS-IS interfaces";
      }
    }

    grouping isis-if-topologies-cfg {
      description "IS-IS interface topology configuration.";
      container metric {
        uses metric-cfg;
        container level-1 {
          uses metric-cfg;
          description "level-1 specific configuration";
        }
        container level-2 {
          uses metric-cfg;
          description "level-2 specific configuration";
        }
        description "Metric IS-IS interface configuration.";
      }
    }

    grouping system-counters {
      container system-counters {
        config false;
        list level {
          key level;

          leaf level {
            type level-number;
            description "IS-IS level.";
          }
          leaf corrupted-lsps {
            type uint32;
            description
              "Number of corrupted in-memory LSPs detected.
               LSPs received from the wire with a bad
               checksum are silently dropped and not counted.
```

```
            LSPs received from the wire with parse errors
            are counted by lsp-errors.";
        }
        leaf authentication-type-fails {
          type uint32;
          description
             "Number of authentication type mismatches.";
        }
        leaf authentication-fails {
          type uint32;
          description
            "Number of authentication key failures.";
        }
        leaf database-overload {
          type uint32;
          description
            "Number of times the database has become
             overloaded.";
        }
        leaf own-lsp-purge {
          type uint32;
          description
            "Number of times a zero-aged copy of the system's
             own LSP is received from some other IS-IS node.";
        }
        leaf manual-address-drop-from-area {
          type uint32;
          description
            "Number of times a manual address
             has been dropped from the area.";
        }
        leaf max-sequence {
          type uint32;
          description
            "Number of times the system has attempted
             to exceed the maximum sequence number.";
        }
        leaf sequence-number-skipped {
          type uint32;
          description
            "Number of times a sequence number skip has
             occured.";
        }
        leaf id-len-mismatch {
          type uint32;
          description
            "Number of times a PDU is received with a
             different value for the ID field length
```

```
                than that of the receiving system.";
            }
            leaf partition-changes {
              type uint32;
              description
                "Number of partition changes detected.";
            }
            leaf lsp-errors {
              type uint32;
              description
                "Number of LSPs with errors we have received.";
            }
            leaf spf-runs {
              type uint32;
              description
                "Number of times we ran SPF at this level.";
            }
            description
              "List of supported levels.";
          }
          description
            "List counters for the IS-IS protocol instance";
        }
        description "System counters grouping.";
      }

      grouping event-counters {
        container event-counters {
          config false;
          leaf adjacency-changes {
            type uint32;
            description
              "The number of times an adjacency state change has
               occured on this interface.";
          }
          leaf adjacency-number {
            type uint32;
            description
              "The number of adjacencies on this interface.";
          }
          leaf init-fails {
            type uint32;
            description
              "The number of times initialization of this
               interface has failed. This counts events such
               as PPP NCP failures. Failures to form an
               adjacency are counted by adjacency-rejects.";
          }
```

```
        leaf adjacency-rejects {
          type uint32;
          description
            "The number of times an adjacency has been
             rejected on this interface.";
        }
        leaf id-len-mismatch {
          type uint32;
          description
            "The number of times an IS-IS PDU with an ID
             field length different from that for this
             system has been received on this interface.";
        }
        leaf max-area-addresses-mismatch {
          type uint32;
          description
            "The number of times an IS-IS PDU has been
             received on this interface with the
             max area address field differing from that of
             this system.";
        }
        leaf authentication-type-fails {
          type uint32;
          description
            "Number of authentication type mismatches.";
        }
        leaf authentication-fails {
          type uint32;
          description
            "Number of authentication key failures.";
        }
        leaf lan-dis-changes {
          type uint32;
          description
            "The number of times the DIS has changed on this
             interface at this level. If the interface type is
             point-to-point,the count is zero.";
        }
        description "IS-IS interface event counters.";
      }
      description
        "Grouping for IS-IS interface event counters";
    }

    grouping packet-counters {
      container packet-counters {
        config false;
        list level {
```

```
            key level;

            leaf level {
              type level-number;
              description "IS-IS level.";
            }
            container iih {
              leaf in {
                type uint32;
                description "Received IIH PDUs.";
              }
              leaf out {
                type uint32;
                description "Sent IIH PDUs.";
              }
              description "Number of IIH PDUs received/sent.";
            }
            container ish {
              leaf in {
                type uint32;
                description "Received ISH PDUs.";
              }
              leaf out {
                type uint32;
                description "Sent ISH PDUs.";
              }
              description
                "ISH PDUs received/sent.";
            }
            container esh {
              leaf in {
                type uint32;
                description "Received ESH PDUs.";
              }
              leaf out {
                type uint32;
                description "Sent ESH PDUs.";
              }
              description "Number of ESH PDUs received/sent.";
            }
            container lsp {
              leaf in {
                type uint32;
                description "Received LSP PDUs.";
              }
              leaf out {
                type uint32;
                description "Sent LSP PDUs.";
```

```
            }
            description "Number of LSP PDUs received/sent.";
          }
          container psnp {
            leaf in {
              type uint32;
              description "Received PSNP PDUs.";
            }
            leaf out {
              type uint32;
              description "Sent PSNP PDUs.";
            }
            description "Number of PSNP PDUs received/sent.";
          }
          container csnp {
            leaf in {
              type uint32;
              description "Received CSNP PDUs.";
            }
            leaf out {
              type uint32;
              description "Sent CSNP PDUs.";
            }
            description "Number of CSNP PDUs received/sent.";
          }
          container unknown {
            leaf in {
              type uint32;
              description "Received unknown PDUs.";
            }
            leaf out {
              type uint32;
              description "Sent unknown PDUs.";
            }
            description "Number of unknown PDUs received/sent.";
          }
          description
            "List of packet counter for supported llevels.";
        }
        description "Packet counters per IS-IS level.";
      }
      description
        "Grouping for per IS-IS Level packet counters.";
    }

    grouping spf-log {
      container spf-log {
        config false;
```

```
          list event {
            key id;

            leaf id {
              type uint32;
              description
                "Event identifier -  purely internal value.";
            }
            leaf spf-type {
              type enumeration {
                enum full {
                  description "Full SPF computation.";
                }
                enum route-only {
                  description
                    "Route reachability only SPF computation";
                }
              }
              description "Type of SPF computation performed.";
            }
            leaf level {
              type level-number;
              description
                "IS-IS level number for SPF computation";
            }
            leaf schedule-timestamp {
              type yang:timestamp;
              description
                "Timestamp of when the SPF computation was
                 scheduled.";
            }
            leaf start-timestamp {
              type yang:timestamp;
              description
                "Timestamp of when the SPF computation started.";
            }
            leaf end-timestamp {
              type yang:timestamp;
              description
                "Timestamp of when the SPF computation ended.";
            }
            list trigger-lsp {
              key "lsp";
              leaf lsp {
                type lsp-id;
                description
                  "LSPID of the LSP triggering SPF computation.";
              }
```

```
            leaf sequence {
              type uint32;
              description
                "Sequence number of the LSP triggering SPF
                 computation";
            }
            description
              "This list includes the LSPs that triggered the
               SPF computation.";
          }
          description
            "List of computation events - implemented as a
             wrapping buffer.";
        }

        description
          "This container lists the SPF computation events.";
      }
      description "Grouping for spf-log events.";
    }

    grouping lsp-log {
      container lsp-log {
        config false;
        list event {
          key id;

          leaf id {
            type uint32;
            description
              "Event identifier -  purely internal value.";
          }
          leaf level {
            type level-number;
            description
              "IS-IS level number for LSP";
          }
          container lsp {
            leaf lsp {

              type lsp-id;
              description
                "LSPID of the LSP.";
            }
            leaf sequence {
              type uint32;
              description
                "Sequence number of the LSP.";
```

```
            }
            description
              "LSP identification container - either the received
               LSP or the locally generated LSP.";
          }

          leaf received-timestamp {
            type yang:timestamp;

            description
              "Timestamp of when the LSP was received. In case
               of local LSP update, the timestamp refers to the
               local LSP update time.";
          }

          leaf change {
            type identityref {
              base lsp-log-reason;
            }
            description "Type of LSP change.";
          }

          description
            "List of LSP events - implemented as a
             wrapping buffer.";
        }

        description
          "LSP reception and local LSP origination events
           container.";

      }
      description "Grouping for LSP log.";
    }

    grouping hostname-db {
      container hostnames {
        config false;
        list hostname {
          key system-id;
          leaf system-id {
            type system-id;
            description
              "System-id associated with the hostname.";
          }
          leaf hostname {
            type string;
            description
```

```
              "Hostname associated with the system ID.";
          }
          description
            "List of system-id/hostname associations.";
        }
        description
          "Hostname to system-id mapping database.";
      }
      description
        "Grouping for hostname to system-id mapping database.";
    }

    /* Groupings for the LSDB description */

    grouping prefix-reachability-attributes {
      description
        "Grouping for extended reachability attributes of an
         IPv4 or IPv6 prefix.";

      leaf external-prefix-flag {
        type boolean;
        description "External prefix flag.";
      }
      leaf readvertisement-flag {
        type boolean;
        description "Readvertisement flag.";
      }
      leaf node-flag {
        type boolean;
        description "Node flag.";
      }
    }

    grouping prefix-ipv4-source-router-id {
      description
        "Grouping for the IPv4 source router ID of a prefix
         advertisement.";

      leaf ipv4-source-router-id {
        type inet:ipv4-address;
        description "IPv4 Source router ID address.";
      }
    }

    grouping prefix-ipv6-source-router-id {
      description
        "Grouping for the IPv6 source router ID of a prefix
         advertisement.";
```

```
        leaf ipv6-source-router-id {
          type inet:ipv6-address;
          description "IPv6 Source router ID address.";
        }
      }

      grouping prefix-attributes-extension {
        description "Prefix extended attributes.";

        uses prefix-reachability-attributes;
        uses prefix-ipv4-source-router-id;
        uses prefix-ipv6-source-router-id;
      }

      grouping prefix-ipv4-std {
        description
          "Grouping for attributes of an IPv4 standard prefix.";
        leaf up-down {
          type boolean;
          description "Value of up/down bit.";
        }
        leaf i-e {
          type boolean;
          description "Value of I/E bit.";
        }
        leaf ip-prefix {
          type inet:ipv4-address;
          description "IPv4 prefix address";
        }
        leaf prefix-len {
          type uint8;
          description "IPv4 prefix length (in bits)";
        }
        leaf default-metric {
          type std-metric;
          description "Default IS-IS metric for IPv4 prefix";
        }
        container delay-metric {
          leaf metric {
            type std-metric;
            description "IS-IS delay metric for IPv4 prefix";
          }
          leaf supported {
            type boolean;
            default "false";
            description
              "Indicates whether IS-IS delay metric is supported.";
          }
```

```
        description "IS-IS delay metric container.";
      }
      container expense-metric {
        leaf metric {
          type std-metric;
          description "IS-IS expense metric for IPv4 prefix";
        }
        leaf supported {
          type boolean;
          default "false";
          description
            "Indicates whether IS-IS delay metric is supported.";
        }
        description "IS-IS expense metric container.";
      }
      container error-metric {
        leaf metric {
          type std-metric;
          description
            "This leaf describes the IS-IS error metric value";
        }
        leaf supported {
          type boolean;
          default "false";
          description "IS-IS error metric for IPv4 prefix";
        }
        description "IS-IS error metric container.";
      }
    }

    grouping prefix-ipv4-extended {
      description
        "Grouping for attributes of an IPv4 extended prefix.";
      leaf up-down {
        type boolean;
        description  "Value of up/down bit.";
      }
      leaf ip-prefix {
        type inet:ipv4-address;
        description "IPv4 prefix address";
      }
      leaf prefix-len {
        type uint8;
        description "IPv4 prefix length (in bits)";
      }

      leaf metric {
        type wide-metric;
```

```
        description "IS-IS wide metric value";
      }
      leaf-list tag {
        type uint32;
        description
          "List of 32-bit tags associated with the IPv4 prefix.";
      }
      leaf-list tag64 {
        type uint64;
        description
          "List of 32-bit tags associated with the IPv4 prefix.";
      }
      uses prefix-attributes-extension;
    }

    grouping prefix-ipv6-extended {
      description "Grouping for attributes of an IPv6 prefix.";
      leaf up-down {
        type boolean;
        description "Value of up/down bit.";
      }
      leaf ip-prefix {
        type inet:ipv6-address;
        description "IPv6 prefix address";
      }
      leaf prefix-len {
        type uint8;
        description  "IPv4 prefix length (in bits)";
      }
      leaf metric {
        type wide-metric;
        description  "IS-IS wide metric value";
      }
      leaf-list tag {
        type uint32;
        description
          "List of 32-bit tags associated with the IPv4 prefix.";
      }
      leaf-list tag64 {
        type uint64;
        description
          "List of 32-bit tags associated with the IPv4 prefix.";
      }
      uses prefix-attributes-extension;
    }

    grouping neighbor-extended {
      description
```

```
        "Grouping for attributes of an IS-IS extended neighbor.";
      leaf neighbor-id {
        type system-id;
        description "System-id of the extended neighbor.";
      }
      leaf metric {
        type wide-metric;
        description "IS-IS wide metric for extended neighbor";
      }
    }

    grouping neighbor {
      description  "IS-IS standard neighbor grouping.";
      leaf neighbor-id {
        type system-id;
        description "IS-IS neighbor system-id";
      }
      leaf i-e {
        type boolean;
        description
          "Internal or External (I/E) Metric bit value";
      }
      leaf default-metric {
        type std-metric;
        description "IS-IS default metric value";
      }
      container delay-metric {
        leaf metric {
          type std-metric;
          description "IS-IS delay metric value";
        }
        leaf supported {
          type boolean;
          default "false";
          description "IS-IS delay metric supported";
        }
        description "IS-IS delay metric container";
      }
      container expense-metric {
        leaf metric {
          type std-metric;
          description "IS-IS delay expense metric value";
        }
        leaf supported {
          type boolean;
          default "false";
          description "IS-IS delay expense metric supported";
        }
```

```
        description "IS-IS delay expense metric container";
      }
      container error-metric {
        leaf metric {
          type std-metric;
          description "IS-IS error metric value";
        }
        leaf supported {
          type boolean;
          default "false";
          description "IS-IS error metric supported";
        }
        description "IS-IS error metric container";
      }
    }

    grouping lsp-entry {
      description "IS-IS LSP database entry groupting";

      leaf decoded-completed {
        type boolean;
        description "IS-IS LSP body fully decoded.";
      }
      leaf raw-data {
        type yang:hex-string;
        description
          "The hexadecial representation of the complete LSP in
           network-byte order (NBO) as received or originated.";
      }
      leaf lsp-id {
        type lsp-id;
        description "LSP ID of the LSP";
      }
      leaf checksum {
        type uint16;
        description "LSP checksum";
      }
      leaf remaining-lifetime {
        type uint16;
        units "seconds";
        description
          "Remaining lifetime (in seconds) until LSP expiration.";
      }
      leaf sequence {
        type uint32;
        description
          "This leaf describes the sequence number of the LSP.";
      }
```

```
        leaf attributes {
          type bits {
            bit partitioned {
              description "Originator partition repair supported";
            }
            bit attached-error {
              description
                "If set, the originator is attached to
                 another area using the referred metric.";
            }
            bit attached-expense {
              description
                "If set, the originator is attached to
                 another area using the referred metric.";
            }
            bit attached-delay {
              description
                "If set, the originator is attached to
                 another area using the referred metric.";
            }
            bit attached-default {
              description
                "If set, the originator is attached to
                 another area using the referred metric.";
            }
            bit overload {
              description
                "If set, the originator is overloaded,
                 and must be avoided in path calculation.";
            }
          }
          description "LSP attributes";
        }

        leaf-list ipv4-addresses {
          type inet:ipv4-address;
          description
            "List of IPv4 addresses of the IS-IS node - IS-IS
             reference is TLV 132.";
        }

        leaf-list ipv6-addresses {
          type inet:ipv6-address;
          description
            "List of IPv6 addresses of the IS-IS node - IS-IS
             reference is TLV 232.";
        }
```

```
        leaf ipv4-te-routerid {
          type inet:ipv4-address;
          description
            "IPv4 Traffic Engineering router ID of the IS-IS node -
             IS-IS reference is TLV 134.";
        }

        leaf ipv6-te-routerid {
          type inet:ipv6-address;
          description
            "IPv6 Traffic Engineering router ID of the IS-IS node -
             IS-IS reference is TLV 140.";
        }

        leaf-list protocol-supported {
          type uint8;
          description
          "List of supported protocols of the IS-IS node -
           IS-IS reference is TLV 129.";
        }

        leaf dynamic-hostname {
          type string;
          description
            "Host Name of the IS-IS node - IS-IS reference
             is TLV 137.";
        }

        container authentication {
          leaf authentication-type {
            type string;
            description
              "Authentication type to be used with IS-IS node.";
          }
          leaf authentication-key {
            type string;
            description
              "Authentication keyto be used. For security reasons,
               the authentication key MUST NOT be presented in
               plaintext format. It is recommended to use an MD5
               hash to present the authentication-key.";
          }
          description
            "IS-IS node authentication information container -
             IS-IS reference is TLV 10.";
        }

        container mt-entries {
```

```
          list topology {
            description
              "List of topologies supported";

            leaf MT-ID {
              type uint16 {
                range "0 .. 4095";
              }
              description
                "Multi-Topolgoy identifier of topology.";
            }

            leaf attributes {
              type bits {
                bit overload {
                  description
                    "If set, the originator is overloaded,
                     and must be avoided in path calculation.";
                }
                bit attached {
                  description
                    "If set, the originator is attached to
                    another area using the referred metric.";
                }
              }
              description
                "Attributes of the LSP for the associated
                 topology.";
            }
          }
          description
            "IS-IS node topology information container -
             IS-IS reference is TLV 229.";
        }

        list router-capabilities {
          leaf flags {
            type bits {
              bit flooding {
                position 0;
                description
                  "If the S bit is set, the IS-IS Router CAPABILITY
                   TLV MUST be flooded across the entire routing
                   domain. If the S bit is clear, the TLV MUST NOT
                   be leaked between levels. This bit MUST NOT
                    be altered during the TLV leaking.";
              }
              bit down {
```

```
            position 1;
            description
              "When the IS-IS Router CAPABILITY TLV is leaked
               from level-2 to level-1, the D bit MUST be set.
               Otherwise, this bit MUST be clear.  IS-IS Router
               capability TLVs with the D bit set MUST NOT be
               leaked from level-1 to level-2 in to prevent
               TLV looping.";
          }
        }
        description "Router Capability Flags";
      }
      container node-tags {
        if-feature node-tag;
        list node-tag {
          leaf tag {
            type uint32;
            description "Node tag value.";
          }
          description "List of tags.";
        }
        description "Node Tag container";
      }

      leaf binary {
        type binary;
        description
          "Binary encoding of the IS-IS node capabilities";
      }
      description
        "IS-IS node capabilities container. This container may
         be extended with detailed information -  IS-IS
         reference is TLV 242.";
    }

    container is-neighbor {
      list neighbor {
        uses neighbor;
        description "List of neighbors.";
      }
      description
        "Standard IS neighbors container - IS-IS reference is
         TLV 2.";
    }

    container extended-is-neighbor {
      list neighbor {
        uses neighbor-extended;
```

```
            description
              "List of extended IS neighbors";
          }
          description
            "Standard IS extended neighbors container - IS-IS
             reference is TLV 22";
        }

        container ipv4-internal-reachability {
          list prefixes {
            uses prefix-ipv4-std;
            description "List of prefixes.";
          }
          description
          "IPv4 internal reachability information container - IS-IS
           reference is TLV 128.";
        }

        container ipv4-external-reachability {
          list prefixes {
            uses prefix-ipv4-std;
            description "List of prefixes.";
          }
          description
            "IPv4 external reachability information container -
             IS-IS reference is TLV 130.";
        }

        container extended-ipv4-reachability {
          list prefixes {
            uses prefix-ipv4-extended;
            description "List of prefixes.";
          }
          description
            "IPv4 extended reachability information container -
             IS-IS reference is TLV 135.";
        }

        container mt-is-neighbor {
          list neighbor {
            leaf mt-id {
              type uint16 {
                range "0 .. 4095";
              }
              description "Multi-topology (MT) identifier";
            }
            uses neighbor-extended;
            description "List of neighbors.";
```

```
        }
        description
          "IS-IS multi-topology neighbor container - IS-IS
           reference is TLV 223.";
      }

      container mt-extended-ipv4-reachability {
        list prefixes {
          leaf mt-id {
            type uint16 {
              range "0 .. 4095";
            }
            description  "Multi-topology (MT) identifier";
          }
          uses prefix-ipv4-extended;
          description "List of extended prefixes.";
        }
        description
          "IPv4 multi-topolgy (MT) extended reachability
           information container - IS-IS reference is TLV 235.";
      }

      container mt-ipv6-reachability {
        list prefixes {
          leaf MT-ID {
            type uint16 {
              range "0 .. 4095";
            }
            description "Multi-topology (MT) identifier";
          }
          uses prefix-ipv6-extended;
          description "List of IPv6 extended prefixes.";
        }
        description
          "IPv6 multi-topolgy (MT) extended reachability
           information container - IS-IS reference is TLV 237.";
      }

      container ipv6-reachability {
        list prefixes {
          uses prefix-ipv6-extended;
          description "List of IPv6 prefixes.";
        }
        description
          "IPv6 reachability information container - IS-IS
           reference is TLV 236.";
        }
      }
```

```
     grouping lsdb {
       description "Link State Database (LSDB) grouping";
       container database {
         config false;
         list level-db {
           key level;

           leaf level {
             type level-number;
             description "LSDB level number (1 or 2)";
           }
           list lsp {
             key lsp-id;
             uses lsp-entry;
             description "List of LSPs in LSDB";
           }
           description "LSP list for LSDB level container";
         }
         description "IS-IS Link State database container";
       }
     }



     /* Augmentations */

     augment "/rt:routing/"
       +"rt:ribs/rt:rib/rt:routes/rt:route" {
       when "rt:source-protocol = 'isis:isis'" {
         description "IS-IS-specific route attributes.";
       }
       uses route-content;
       description
         "This augments route object in RIB with IS-IS-specific
          attributes.";
     }


     augment "/if:interfaces/if:interface" {
       leaf clns-mtu {
         type uint16;
         description "CLNS MTU of the interface";
       }
       description "ISO interface config.";
     }

     augment "/rt:routing/rt:control-plane-protocols/"
```

```
      +"rt:control-plane-protocol" {
      when "rt:type = 'isis:isis'" {
        description
          "This augment is only valid when routing protocol
           instance type is 'isis'";
      }
      description
        "This augments a routing protocol instance with IS-IS
         specific parameters.";
      container isis {
        must "count(area-address) > 0" {
          error-message
            "At least one area-address must be configured.";
          description
            "Enforce configuration of at least one area.";
        }

        uses isis-global-cfg;
        container fast-reroute {
          if-feature fast-reroute;
          uses fast-reroute-global-cfg;
          uses fast-reroute-global-state;
          description
            "IP Fast ReRoute (IPFRR) global container";
        }
        container spf-control {
          leaf paths {
            if-feature max-ecmp;
            type uint16 {
              range "1..32";
            }
            description
              "Maximum number of Equal-Cost Multi-Path (ECMP) paths.";
          }
          container ietf-spf-delay {
            if-feature ietf-spf-delay;
            uses ietf-spf-delay-cfg;
            uses ietf-spf-delay-state;
            description "IETF SPF delay algorithm container";
          }
          description
            "SPF computation-related information container";
        }
        container topologies {
          if-feature multi-topology;
          list topology {
            key "name";
            leaf enable {
```

```
            type boolean;
            description "Topology enable configuration";
          }
          leaf name {
            type leafref {
              path "../../../../../../rt:ribs/rt:rib/rt:name";
            }
            description
              "Routing Information Base (RIB) corresponding
               to topology.";
          }

          uses isis-global-topologies-cfg;

          description "List of topologies";
        }
        description "Multi-topology container";
      }
      container interfaces {
        list interface {
          key "name";
          leaf name {
            type if:interface-ref;

            description
              "Reference to the interface within
               the routing-instance.";
          }
          uses isis-if-cfg;
          container fast-reroute {
            if-feature fast-reroute;
            uses fast-reroute-if-cfg;
            description
              "IP Fast ReRoute (IPFRR) interface container";
          }
          container topologies {
            if-feature multi-topology;
            list topology {
              key name;

              leaf name {
                type leafref {
                  path "../../../../../../../../"+
                    "rt:ribs/rt:rib/rt:name";
                }

                description
                  "Routing Information Base (RIB) corresponding
```

```
                    to topology.";
                }
                uses isis-if-topologies-cfg;
                description "List of interface topologies";
              }
              description "Multi-topology container";
            }
            uses adjacency-state;
            uses event-counters;
            uses packet-counters;
            description "List of IS-IS interfaces.";
          }
          description
            "IS-IS interface specific configuration container";
        }
        uses spf-log;
        uses lsp-log;
        uses hostname-db;
        uses lsdb;
        uses local-rib;
        uses system-counters;

        description
          "IS-IS configuration/state top-level container";
      }
    }


    /* RPC methods */

    rpc clear-adjacency {
      description
        "This RPC request clears a particular set of IS-IS
         adjacencies. If the operation fails due to an internal
         reason, then the error-tag and error-app-tag should be
         set indicating the reason for the failure.";
      input {

        leaf routing-protocol-instance-name {
          type instance-state-ref;
          mandatory "true";
          description
            "Name of the IS-IS protocol instance whose IS-IS
             information is being queried.

             If the corresponding IS-IS instance doesn't exist,
             then the operation will fail with an error-tag of
             'data-missing' and an error-app-tag of
```

```
            'routing-protocol-instance-not-found'.";
        }
        leaf level {
          type level;
          description
            "IS-IS level of the adjacency to be cleared. If the
             IS-IS level is level-1-2, both level 1 and level 2
             adjacencies would be cleared.

             If the value provided is different from the one
             authorized in the enum type, then the operation
             SHALL fail with an error-tag of 'data-missing' and
             an error-app-tag of 'bad-isis-level'.";
        }
        leaf interface {
          type string;
          description
            "IS-IS interface name.

             If the corresponding IS-IS interface doesn't exist,
             then the operation SHALL fail with an error-tag of
             'data-missing' and an error-app-tag of
             'isis-interface-not-found'.";
        }
      }
    }

    rpc clear-database {
      description
        "This RPC request clears a particular  IS-IS database. If
         the operation fails for an IS-IS internal reason, then
         the error-tag and error-app-tag should be set
         indicating the reason for the failure.";
      input {
        leaf routing-protocol-instance-name {
          type instance-state-ref;
          mandatory "true";
          description
            "Name of the IS-IS protocol instance whose IS-IS
             database(s) is/are being cleard.

             If the corresponding IS-IS instance doesn't exist,
             then the operation will fail with an error-tag of
             'data-missing' and an error-app-tag of
             'routing-protocol-instance-not-found'.";
        }
        leaf level {
          type level;
```

```
          description
            "IS-IS level of the adjacency to be cleared. If the
             IS-IS level is level-1-2, both level 1 and level 2
             databases would be cleared.

             If the value provided is different from the one
             authorized in the enum type, then the operation
             SHALL fail with an error-tag of 'data-missing' and
             an error-app-tag of 'bad-isis-level'.";
        }
      }
    }


    /* Notifications */

    notification database-overload {
      uses notification-instance-hdr;

      leaf overload {
        type enumeration {
          enum off {
            description
              "Indicates IS-IS instance has left overload state";
          }
          enum on {
            description
              "Indicates IS-IS instance has entered overload state";
          }

        }
        description "New overload state of the IS-IS instance";
      }
      description
        "This notification is sent when an IS-IS instance
         overload state changes.";
    }

    notification lsp-too-large {
      uses notification-instance-hdr;
      uses notification-interface-hdr;

      leaf pdu-size {
        type uint32;
        description "Size of the LSP PDU";
      }
      leaf lsp-id {
        type lsp-id;
```

```
        description "LSP ID";
      }
      description
        "This notification is sent when we attempt to propagate
         an LSP that is larger than the dataLinkBlockSize for the
         circuit.  The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification if-state-change {
      uses notification-instance-hdr;
      uses notification-interface-hdr;

      leaf state {
        type if-state-type;
        description "Interface state.";
      }
      description
        "This notification is sent when an interface
         state change is detected.";
    }

    notification corrupted-lsp-detected {
      uses notification-instance-hdr;
      leaf lsp-id {
        type lsp-id;
        description "LSP ID";
      }
      description
        "This notification is sent when we find that
         an LSP that was stored in memory has become
         corrupted.";
    }

    notification attempt-to-exceed-max-sequence {
      uses notification-instance-hdr;
      leaf lsp-id {
        type lsp-id;
        description "LSP ID";
      }
      description
        "This notification is sent when the system
         wraps the 32-bit sequence counter of an LSP.";
    }

    notification id-len-mismatch {
      uses notification-instance-hdr;
```

```
      uses notification-interface-hdr;

      leaf pdu-field-len {
        type uint8;
        description "Size of the ID length in the received PDU";
      }
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when we receive a PDU
         with a different value for the System ID length.
         The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification max-area-addresses-mismatch {
      uses notification-instance-hdr;
      uses notification-interface-hdr;

      leaf max-area-addresses {
        type uint8;
        description "Received number of supported areas";
      }
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when we receive a PDU
         with a different value for the Maximum Area Addresses.
         The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification own-lsp-purge {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf lsp-id {
        type lsp-id;
        description "LSP ID";
      }
      description
        "This notification is sent when the system receives
         a PDU with its own system ID and zero age.";
```

```
    }

    notification sequence-number-skipped {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf lsp-id {
        type lsp-id;
        description "LSP ID";
      }
      description
        "This notification is sent when the system receives a
         PDU with its own system ID and different contents. The
         system has to reoriginate the LSP with a higher sequence
         number.";
    }

    notification authentication-type-failure {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when the system receives a
         PDU with the wrong authentication type field.
         The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification authentication-failure {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when the system receives
         a PDU with the wrong authentication information.
         The notification generation must be throttled with
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification version-skew {
      uses notification-instance-hdr;
```

```
      uses notification-interface-hdr;
      leaf protocol-version {
        type uint8;
        description "Protocol version received in the PDU.";
      }
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when the system receives a
         PDU with a different protocol version number.
         The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification area-mismatch {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      description
        "This notification is sent when the system receives a
         Hello PDU from an IS that does not share any area
         address. The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification rejected-adjacency {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf raw-pdu {
        type binary;
        description
          "Received raw PDU.";
      }
      leaf reason {
        type string;
        description
          "The system may provide a reason to reject the
           adjacency. If the reason is not available,
           an empty string will be returned.";
      }
      description
```

```
        "This notification is sent when the system receives a
         Hello PDU from an IS but does not establish an adjacency
         for some reason. The notification generation must be
         throttled with at least 5 seconds betweeen successive
         notifications.";
    }


    notification protocols-supported-mismatch {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      leaf-list protocols {
        type uint8;
        description
          "List of protocols supported by the remote system.";
      }
      description
        "This notification is sent when the system receives a
         non-pseudonode LSP that has no matching protocols
         supported. The notification generation must be throttled
         with at least 5 seconds betweeen successive
         notifications.";
    }

    notification lsp-error-detected {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf lsp-id {
        type lsp-id;
        description "LSP ID.";
      }
      leaf raw-pdu {
        type binary;
        description "Received raw PDU.";
      }
      leaf error-offset {
        type uint32;
        description
          "If the problem is a malformed TLV, the error-offset
           points to the start of the TLV. If the problem is with
           the LSP header, the error-offset points to the errant
           byte";
      }
      leaf tlv-type {
```

```
        type uint8;
        description
          "If the problem is a malformed TLV, the tlv-type is set
           to the type value of the suspicious TLV. Otherwise,
           this leaf is not present.";
      }
      description
        "This notification is sent when the system  receives an
   '     LSP with a parse error. The notification generation must
         be throttled with at least 5 seconds betweeen successive
         notifications.";
    }

    notification adjacency-state-change {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
      leaf neighbor {
        type string;
        description
          "Name of the neighbor. If the name of the neighbor is
           not available, it is not returned.";
      }
      leaf neighbor-system-id {
        type system-id;
        description "Neighbor system-id";
      }
      leaf state {
        type adj-state-type;

        description "New state of the IS-IS adjacency.";
      }
      leaf reason {
        type string;
        description
          "If the adjacency is going to DOWN,  this leaf provides
           a reason for the adjacency going down. The reason is
           provided as a text. If the adjacency is going to UP, no
           reason is provided.";
      }
      description
        "This notification is sent when an IS-IS adjacency
         moves to Up state or to Down state.";
    }

    notification lsp-received {
      uses notification-instance-hdr;
      uses notification-interface-hdr;
```

```
       leaf lsp-id {
         type lsp-id;
         description "LSP ID";
       }
       leaf sequence {
         type uint32;
         description "Sequence number of the received LSP.";
       }
       leaf received-timestamp {
         type yang:timestamp;

         description "Timestamp when the LSP was received.";
       }
       leaf neighbor-system-id {
         type system-id;
         description "Neighbor system-id of LSP sender";
       }
       description
         "This notification is sent when an LSP is received.
          The notification generation must be throttled with at
          least 5 seconds betweeen successive notifications.";
     }

     notification lsp-generation {
       uses notification-instance-hdr;

       leaf lsp-id {
         type lsp-id;
         description "LSP ID";
       }
       leaf sequence {
         type uint32;
         description "Sequence number of the received LSP.";
       }
       leaf send-timestamp {
         type yang:timestamp;

         description "Timestamp when our LSP was regenerated.";
       }
       description
         "This notification is sent when an LSP is regenerated.
          The notification generation must be throttled with at
          least 5 seconds betweeen successive notifications.";
     }
   }

   <CODE ENDS>
```

7. Security Considerations

Configuration and state data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241].

As IS-IS is an IGP protocol (critical piece of the network), ensuring stability and security of the protocol is mandatory for the network service.

Authors recommends to implement NETCONF access control model ([RFC6536]) to restrict access to all or part of the configuration to specific users. Access control to RPCs is also critical as RPC allows to clear protocol datastructures that would definitively impact the network service. This kind of RPC needs only to be used in specific cases by well-known experienced users.

Authors consider that all the configuration is considered as sensitive/vulnerable as well as RPCs. But security teams can decide to open some part of the configuration to less experienced users depending on the internal organization, for example:

- User FullWrite: would access to the whole data model. This kind of profile may be restricted to few experienced people.
- User PartialWrite: would only access to configuration part within /isis/interfaces/interface. So this kind of profile is restricted to creation/modification/deletion of interfaces. This profile does not have access to RPC.
- User Read: would only access to the operational states.

Unauthorized access to configuration or RPC may cause high damages to the network service.

The "isis/database" may contain authentication information. As presented in the description of the "/isis/database/level-1/lsp/authentication/authentication-key", the authentication MUST never be displayed in a plaintext format for security reason. Authors recommend the usage of MD5 to display or return the authentication-key.

Some authentication-key may also be required in the "isis" writable container. When configuring IS-IS using the NETCONF protocol, authors recommends the usage of secure transport of NETCONF using SSH ([RFC6242]).

8. Contributors

Authors would like to thank Kiran Agrahara Sreenivasa, Dean Bogdanovic, Yingzhen Qu, Yi Yang for their major contributions to the draft.

9. Acknowledgements

TBD.

10. IANA Considerations

The IANA is requested to assign two new URIs from the IETF XML registry ([RFC3688]). Authors are suggesting the following URI:

```
URI: urn:ietf:params:xml:ns:yang:ietf-isis
Registrant Contact: IS-IS WG
XML: N/A, the requested URI is an XML namespace
```

This document also requests one new YANG module name in the YANG Module Names registry ([RFC6020]) with the following suggestion:

```
name: ietf-isis
namespace: urn:ietf:params:xml:ns:yang:ietf-isis
prefix: isis
reference: RFC XXXX
```

11. Change log for ietf-isis YANG module

11.1. From version -22 to version -24

- Fix revision date of the module

11.2. From version -21 to version -22

- TE router-id modeling alignment with OSPF.
- Add max-ecmp + feature in spf-control container (alignment with OSPF).

11.3. From version -20 to version -21

- Model revision date fix

11.4. From version -19 to version -20

- Moved to Yang 1.1
- Lower case enumerations
- Add RFC references to features
- Remove segment-routing feature
- Modified BFD activation modeling

11.5. From version -18 to version -19

- Align with draft-ietf-netmod-rfc8022bis.
- Modify address family types as per draft-ietf-rtgwg-routing-types-17.

11.6. From version -17 to version -18

- NMDA compliancy.
- Set some default values.
- Align with iana-rt-types module.

11.7. From version -16 to version -17

- Cosmetic fixes.
- Use of rt-types model.

11.8. From version -15 to version -16

- Alignment with last IETF key chain model.
- lsp-log "change" leaf moved as an identity.
- Incremental SPF removed from spf-log types.

11.9. From version -14 to version -15

- Alignment with OSPF model done:
  * Added spf-control container with IETF SPF delay algorithm as a feature.

* Added graceful-restart options.
* Added nsr as a feature.
* Removed per topology FRR. Need to be augmented if necessary.
* Created an ldp container within mpls.
* Renamed igp-ldp-sync to igp-sync.
* Added auto-cost container.
* Moved reference-bandwidth under auto-cost container.
* Added IS-IS local RIB as operational state.
* Added decode-completed and raw-data leaves in the LSDB model.
* Modified the notification header.

11.10. From version -13 to version -14

- Segment Routing extensions are now in a separate document.

11.11. From version -12 to version -13

- Move feature nlpid-control to container rather than list.
- Rename multi-topology to topologies to align with OSPF.
- Rename bfd/enabled to bfd/enable for consistency reason.
- Add support for NSR with a feature.

11.12. From version -09 to version -12

- Rename node-tag container to node-tags.

11.13. From version -08 to version -09

- Added container before af list.
- Added container before topology list.
- Aligned LFA if per level cfg.
- Align to draft-ietf-netmod-routing-cfg-23.

11.14. From version -07 to version -08

- Remove selector from system-id type.
- Add some default values.
- Moved lists to containers+groupings for per level configuration.
- remove routing-instance as per core routing model v21.
- added BFD leaf (no more BFD protocol model).
- changed keychain module reference.

11.15. From version -05 to version -07

- Move Overload config from list to container.
- Move Overload-max-metric config from list to container.
- Move preference config from list to container.
- Add Node flag in config.
- Removed BFD config => moved to isis-bfd module.
- Remove call to routing policy model.

11.16. From version -03 to version -05

- Correct invalid references to previous versions of core routing model.
- Remove BFD config and replace by groupings from ietf-bfd.
- Adding routing-policy support through routing-policy model.

11.17. From version -02 to version -03

- Reviewed config and op state groupings.
- Add default value to lfa candidate-disabled.
- Add enable leaf to isis container to reflect admin state.
- Move to VRF centric only.
- Segment routing is part os a separate module.

11.18. From version -01 to version -02

- Adding IPFRR.
- Adding igp-ldp-sync.
- Adding segment-routing.
- Adding instance reference to operational states.
- Move AF type from string to identity.
- Updated router-capability in LSDB description.
- packet counters moved to interface-packet-counters.
- Added modification information in lsp-log.
- Removing igp-ldp-sync timer in IS-IS.
- Defining hierarchy for operational states.
- Adding clns-mtu.
- Adding key-chain.

11.19. From version -00 to version -01

- Interface metric move from af container to interface container.
- Hello-padding on interface moved to hello-padding-disable with empty type.
- three-way-handshake removed.
- route preference changed to a choice.
- csnp-authentication/psnp-authentication merged to authentication container.
- lsp-gen-interval-exp-delay removed.
- Added overload-max-metric feature.
- overload-max-metric is in a separate container.
- Change hello-padding to container.

- Change bfd to container.
- Make BFD a feature.
- Create mpls-te container and put router-id inside.
- Remove GR helper disable and timers.

12. Normative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.

[I-D.ietf-rtgwg-yang-key-chain]
Lindem, A., Qu, Y., Yeung, D., Chen, I., and Z. Zhang, "Routing Key Chain YANG Data Model", draft-ietf-rtgwg-yang-key-chain-24 (work in progress), April 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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <https://www.rfc-editor.org/info/rfc5443>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <https://www.rfc-editor.org/info/rfc7490>.

Appendix A. Example of IS-IS configuration in XML

This section gives an example of configuration of an IS-IS instance on a device. The example is written in XML.

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
      <name>SLI</name>
      <router-id>1.1.1.1</router-id>
      <description/>
      <interfaces>
        <interface>
          <name>Loopback0</name>
        </interface>
                <interface>
          <name>Eth1</name>
        </interface>
      </interfaces>
      <control-plane-protocols>
        <control-plane-protocol>
          <name>ISIS</name>
          <description/>
          <type>isis:isis</type>
          <isis xmlns="urn:ietf:params:xml:ns:yang:ietf-isis">
                          <enable>true</enable>
                          <level-type>level-2</level-type>
              <system-id>87FC.FCDF.4432</system-id>
              <area-address>49.0001</area-address>
```

```
              <mpls-te>
                <ipv4-router-id>1.1.1.1</ipv4-router-id>
              </mpls-te>
              <lsp-lifetime>65535</lsp-lifetime>
              <lsp-refresh>65000</lsp-refresh>
              <metric-type>
                <value>wide</value>
              </metric-type>
              <default-metric>
                <value>111111</value>
              </default-metric>
              <afs>
                              <af>
                                      <af>ipv4-unicast</af>
                                      <enabled>true</enabled>
                              </af>
              </afs>
              <interfaces>
                <interface>
                  <name>Loopback0</name>
                  <tag>200</tag>
                  <metric>
                    <value>0</value>
                  </metric>
                                <passive>true</passive>
                </interface>
                              <interface>
                  <name>Eth1</name>
                                <level-type>level-2</level-type>
                                <interface-type>point-to-point</interface-type
>
                  <metric>
                    <value>167890</value>
                  </metric>
                </interface>
              </interfaces>
            </isis>
          </control-plane-protocol>
        </control-plane-protocols>
    </routing>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>Loopback0</name>
        <description/>
        <type/>
        <link-up-down-trap-enable/>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <mtu/>
          <address>
```

```
          <ip>1.1.1.1</ip>
          <prefix-length>32</prefix-length>
        </address>
      </ipv4>

    </interface>
      <interface>
      <name>Eth1</name>
      <description/>
      <type/>
      <link-up-down-trap-enable/>
      <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <mtu/>
        <address>
          <ip>10.0.0.1</ip>
          <prefix-length>30</prefix-length>
        </address>
      </ipv4>

    </interface>
  </interfaces>
</data>
```

Authors' Addresses

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Derek Yeung
Arrcus, Inc

Email: derek@arrcus.com

Acee Lindem
Cisco Systems

Email: acee@cisco.com

Jeffrey Zhang
Juniper Networks

Email: zzhang@juniper.net

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 16, 2018

P. Psenak, Ed.
Cisco Systems
S. Hegde
Juniper Networks, Inc.
C. Filsfils
K. Talaulikar
Cisco Systems, Inc.
A. Gulko
Thomson Reuters
May 15, 2018

IGP Flexible Algorithm
draft-ietf-lsr-flex-algo-00.txt

Abstract

IGP protocols traditionally compute best paths over the network based on the IGP metric assigned to the links. Many network deployments use RSVP-TE based or Segment Routing based Traffic Engineering to enforce traffic over a path that is computed using different metrics or constraints than the shortest IGP path. This document proposes a solution that allows IGPs themselves to compute constraint based paths over the network. This document also specifies a way of using Segment Routing Prefix-SIDs to steer packets along the constraint-based paths.

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 November 16, 2018.

Table of Contents

1. Introduction

An IGP computed path based on the shortest IGP metric must often be replaced by traffic engineered path due to the traffic requirements which are not reflected by the IGP metric. Some networks engineer the IGP metric assignments in a way that the IGP Metric reflects the link bandwidth or delay. If, for example, the IGP metric is reflecting the bandwidth on the link and the application traffic is delay sensitive, the best IGP path may not reflect the best path from such application's perspective.

To overcome this limitation, various sorts of traffic engineering have been deployed, including RSVP-TE and SR-TE, in which case the TE component is responsible for computing the path based on additional metrics and/or constraints. Such paths need to be installed in the forwarding tables in addition to, or as a replacement for the original paths computed by IGPs. Tunnels are often used to represent the engineered paths and mechanisms like one described in [RFC3906] are used to replace the native IGP paths with such tunnel paths.

This document specifies a set of extensions to ISIS, OSPFv2 and OSPFv3 that enable a router to send TLVs that (a) describe a set of constraints on the topology, (b) identify calculation-type, and (c) metric-type that are to be used to compute the best paths along the constrained topology. A given combination of calculation-type, metric-type and constraints is known as a "Flexible Algorithm Definition". A router that sends such a set of TLVs also assigns a specific value, Flex-Algorithm, to the specified combination of calculation-type, metric-type and constraints.

This document also specifies a way for a router to use IGPs to associate one or more Segment Routing Prefix-SIDs with a particular Flex-Algorithm. Each such Prefix-SID then represents a path that is computed according to the identified Flex-Algorithm.

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 [BCP14] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This section defines terms that are often used in this document.

Flexible Algorithm Definition - the set consisting of (a) calculation-type, (b) metric-type and (c) a set of constraints.

Flexible Algorithm - a numeric identifier in the range 128-255 that is associated via provisioning with the Flexible-Algorithm Definition.

Local Flexible Algorithm Definition - Flexible Algorithm Definition defined locally on the node.

Remote Flexible Algorithm Definition - Flexible Algorithm Definition received from other nodes via IGP flooding.

Flexible Algorithm Participation - per application configuration state that expresses whether the node is participating in a particular Flexible Algorithm.

IGP Algorithm - value from the the "IGP Algorithm Types" registry defined under "Interior Gateway Protocol (IGP) Parameters" IANA registries. IGP Algorithms represents the triplet (Calculation Type, Metric, Constraints), where the second and third elements of the triple MAY not exist.

4. Flexible Algorithm

Many possible constraints may be used to compute a path over a network. Some networks are deployed as multiple planes. A simple form of constraint may be to use a particular plane. A more sophisticated form of constraint can include some extended metric as described in [RFC7810]. Constraints which restrict paths to links with specific affinities or avoid links with specific affinities are also possible. Combinations of these are also possible.

To provide maximum flexibility, we want to provide a mechanism that allows a router to (a) identify a particular calculation-type, (b) metric-type, (c) describe a particular set of constraints, and (d)

assign a numeric identifier, referred to as Flex-Algorithm, to the combination of that calculation-type, metric-type and those constraints. We want the mapping between the Flex-Algorithm and it's meaning to be flexible and defined by the user. As long as all routers in the domain have a common understanding as to what a particular Flex-Algorithm represents, the resulting routing computation is consistent and traffic is not subject to any looping.

The set consisting of (a) calculation-type, (b) metric-type and (c) a set of constraints is referred to as a Flexible-Algorithm Definition.

Flexible-Algorithm is a numeric identifier in the range 128-255 that is associated via provisioning with the Flexible-Algorithm Definition.

IANA "IGP Algorithm Types" registry defines the set of values for IGP Algorithms. We propose to allocate the following values for Flex-Algorithms from this registry:

128-255 - Flex-Algorithms

5. Flexible Algorithm Definition Advertisement

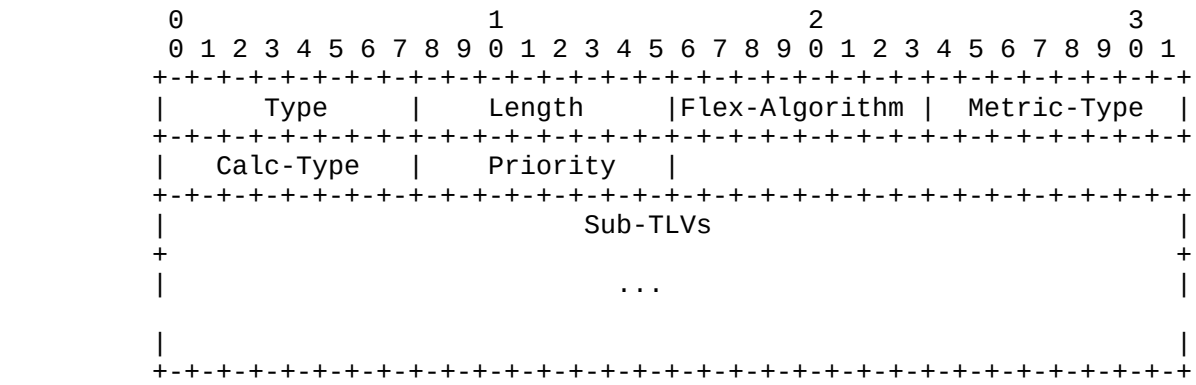
To guarantee the loop free forwarding for paths computed for a particular Flex-Algorithm, all routers that (a) are configured to participate in a particular Flex-Algorithm, and (b) are in the same Flex-Algorithm definition advertisement scope MUST agree on the definition of the Flex-Algorithm.

5.1. ISIS Flexible Algorithm Definition Sub-TLV

ISIS Flexible Algorithm Definition Sub-TLV (FAD Sub-TLV) is used to advertise the definition of the Flex-Algorithm.

ISIS FAD Sub-TLV is advertised as a Sub-TLV of the ISIS Router Capability TLV-242 that is defined in [RFC7981].

ISIS FAD Sub-TLV has the following format:



where:

Type: TBD, suggested value 26

Length: variable, dependent on the included Sub-TLVs

Flex-Algorithm: Single octet value between 128 and 255 inclusive.

Metric-Type: Type of metric to be used during the calculation. Following values are defined:

0: IGP Metric

1: Min Unidirectional Link Delay as defined in [RFC7810].

2: TE default metric as defined in [RFC5305].

Calc-Type: value from 0 to 127 inclusive from the "IGP Algorithm Types" registry defined under "Interior Gateway Protocol (IGP) Parameters" IANA registries. IGP algorithms in the range of 0-127 have a defined triplet (Calculation Type, Metric, Constraints). When used to specify the Calc-Type in the FAD Sub-TLV, only the Calculation Type defined for the specified IGP Algorithm is used. The Metric/Constraints MUST NOT be inherited. If the required calculation type is Shortest Path First, the value 0 SHOULD appear in this field.

Priority: Value between 0 and 255 inclusive that specifies the priority of the advertisement.

Sub-TLVs - optional sub-TLVs.

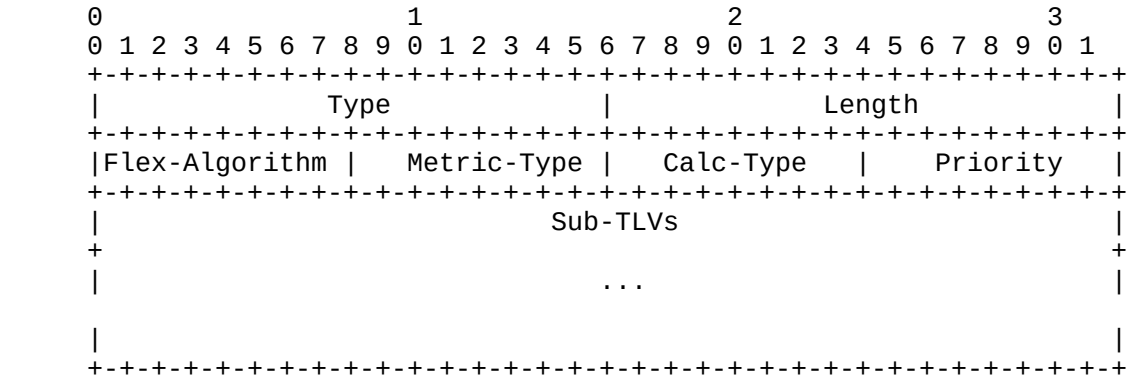
The ISIS FAD Sub-TLV MAY be flooded only in a given level or throughout the domain. In the latter case the S-flag is set as

described in [RFC7981]. It is recommended that domain-wide flooding NOT be the default behavior.

5.2. OSPF Flexible Algorithm Definition TLV

OSPF FAD TLV is advertised as a top-level TLV of the RI LSA that is defined in [RFC7770].

OSPF FAD TLV has the following format:



where:

Type: TBD, suggested value 16

Length: variable, dependent on the included Sub-TLVs

Flex-Algorithm:: Flex-Algorithm number. Value between 128 and 255 inclusive.

Metric-Type: as described in Section 5.1

Calc-Type: as described in Section 5.1

Priority: as described in Section 5.1

Sub-TLVs - optional sub-TLVs.

When multiple OPSF FAD TLVs, for the same Flexible-Algorithm, are received from a given router, the receiver MUST use the first occurrence of the TLV in the Router Information LSA. If the OSPF FAD TLV, for the same Flex-Algorithm, appears in multiple Router Information LSAs that have different flooding scopes, the OSPF FAD TLV in the Router Information LSA with the area-scoped flooding scope

MUST be used. If the OSPF FAD TLV, for the same algorithm, appears in multiple Router Information LSAs that have the same flooding scope, the OSPF FAD TLV in the Router Information (RI) LSA with the numerically smallest Instance ID MUST be used and subsequent instances of the OSPF FAD TLV MUST be ignored.

The RI LSA can be advertised at any of the defined opaque flooding scopes (link, area, or Autonomous System (AS)). For the purpose of OSPF FAD TLV advertisement, area-scoped flooding is REQUIRED. The Autonomous System flooding scope SHOULD not be used by default unless local configuration policy on the originating router indicates domain wide flooding.

5.3. Common Handling of Flexible Algorithm Definition TLV

This section describes the protocol independent handling of the FAD TLV (OSPF) or FAD Sub-TLV (ISIS). We will refer to it as FAD TLV in this section, even though in case of ISIS it is a Sub-TLV.

The value of the Flex-Algorithm MUST be between 128 and 255 inclusive. If it is not, the FAD TLV MUST be ignored.

Not every router configured to participate in a particular Flex-Algorithm need a local definition of such Flex-Algorithm. Only a subset of the routers participating in the particular Flex-Algorithm need the local definition of the Flex-Algorithm.

Every router, that is configured to participate in a particular Flex-Algorithm, MUST select the Flex-Algorithm definition based on the following ordered rules. This allows for the consistent Flex-Algorithm definition selection in cases where different routers advertise different definitions for a given Flex-Algorithm:

1. From the advertisements of the FAD in the area (including both locally generated advertisements and received advertisements) select the one(s) with the highest priority.

2. If there are multiple advertisements of the FAD with the same highest priority, select the one that is originated from the router with the highest System-ID in case of ISIS or Router ID in case of OSPFv2 and OSPFv3. For ISIS the System-ID is described in [ISO10589]. For OSPFv2 and OSPFv3 standard Router ID is described in [RFC2328] and [RFC5340] respectively.

A router that is not configured to participate in a particular Flex-Algorithm MUST ignore FAD Sub-TLVs advertisements for such Flex-Algorithm.

Any change in the Flex-Algorithm definition may result in temporary disruption of traffic that is forwarded based on such Flex-Algorithm paths. The impact is similar to any other event that requires network wide convergence.

If a node is configured to participate in a particular Flexible-Algorithm, but the selected Flex-Algorithm definition includes calculation-type, metric-type or constraint that is not supported by the node, it MUST stop participating in such Flexible-Algorithm. That implies that it MUST NOT announce participation for such Flexible-Algorithm and it MUST remove any forwarding state associated with it.

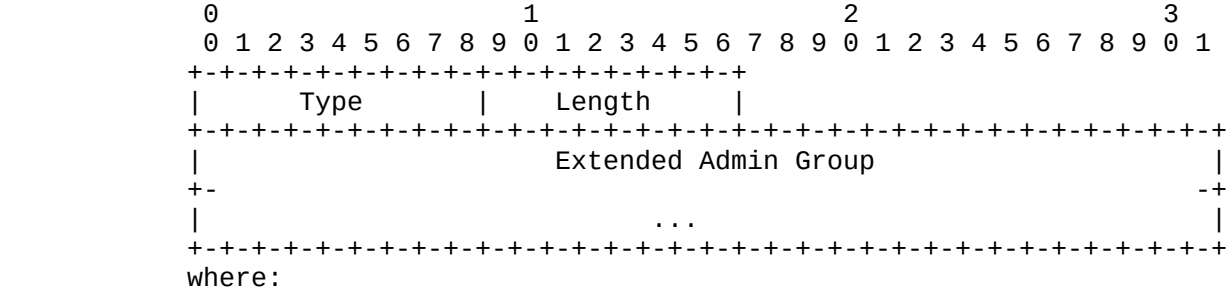
Flex-Algorithm definition is topology independent. It applies to all topologies that a router participates in.

6. Sub-TLVs of ISIS FAD Sub-TLV

6.1. ISIS Flexible Algorithm Exclude Admin Group Sub-TLV

The Flexible-Algorithm definition can specify ’colors’ that are used by the operator to exclude links during the Flex-Algorithm path computation.

Flexible Algorithm Exclude Admin Group Sub-TLV (FAEAG Sub-TLV) is a Sub-TLV of the ISIS FAD Sub-TLV. It has the following format:



where:

Type: 1

Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

Extended Administrative Group: Extended Administrative Group as defined in [RFC7308].

ISIS FAEAG Sub-TLV MAY NOT appear more then once in an ISIS FAD Sub-TLV. If it appears more then once, the ISIS FAD Sub-TLV MUST be ignored by the receiver.

6.2. ISIS Flexible Algorithm Include-Any Admin Group Sub-TLV

The Flexible-Algorithm definition can specify 'colors' that are used by the operator to include link during the Flex-Algorithm path computation.

ISIS Flexible Algorithm Include-Any Admin Group Sub-TLV is used to advertise include-any rule that is used during the Flex-Algorithm path calculation as specified in Section Section 10.

The format of the SIS Flexible Algorithm Include-Any Admin Group Sub-TLV is identical to the format of the FAEAG Sub-TLV in Section 6.1.

Flexible Algorithm Include-Any Admin Group Sub-TLV Type is 2.

ISIS Flexible Algorithm Include-Any Admin Group Sub-TLV MAY NOT appear more then once in an ISIS FAD Sub-TLV. If it appears more then once, the ISIS FAD Sub-TLV MUST be ignored by the receiver.

6.3. ISIS Flexible Algorithm Include-All Admin Group Sub-TLV

The Flexible-Algorithm definition can specify 'colors' that are used by the operator to include link during the Flex-Algorithm path computation.

ISIS Flexible Algorithm Include-All Admin Group Sub-TLV is used to advertise include-all rule that is used during the Flex-Algorithm path calculation as specified in Section Section 10.

The format of the SIS Flexible Algorithm Include-All Admin Group Sub-TLV is identical to the format of the FAEAG Sub-TLV in Section 6.1.

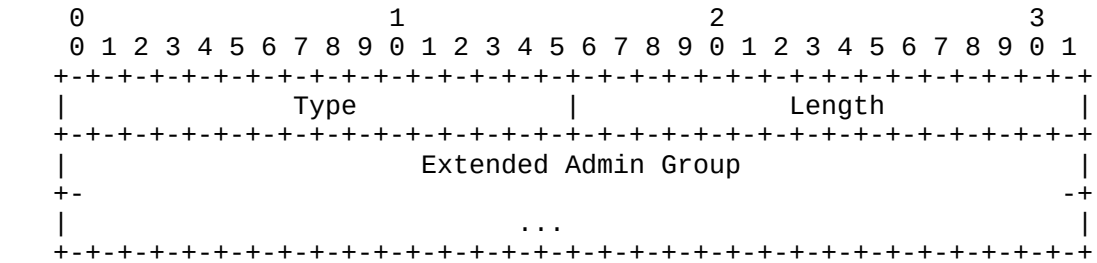
ISIS Flexible Algorithm Include-All Admin Group Sub-TLV Type is 3.

ISIS Flexible Algorithm Include-All Admin Group Sub-TLV MAY NOT appear more then once in an ISIS FAD Sub-TLV. If it appears more then once, the ISIS FAD Sub-TLV MUST be ignored by the receiver.

7. Sub-TLVs of OSPF FAD TLV

7.1. OSPF Flexible Algorithm Exclude Admin Group Sub-TLV

Flexible Algorithm Exclude Admin Group Sub-TLV (FAEAG Sub-TLV) is a Sub-TLV of the OSPF FAD TLV. It's usage is described in Section 6.1. It has the following format:



where:

Type: 1

Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

Extended Administrative Group: Extended Administrative Group as defined in [RFC7308].

OSPF FAEAG Sub-TLV MAY NOT appear more then once in an OSPF FAD TLV. If it appears more then once, the OSPF FAD TLV MUST be ignored by the receiver.

7.2. OSPF Flexible Algorithm Include-Any Admin Group Sub-TLV

The usage of this Sub-TLVs is described in Section 6.2.

The format of the OSPF Flexible Algorithm Include-Any Admin Group Sub-TLV is identical to the format of the OSPF FAEAG Sub-TLV in Section 7.1.

Flexible Algorithm Include-Any Admin Group Sub-TLV Type is 2.

OSPF Flexible Algorithm Include-Any Admin Group Sub-TLV MAY NOT appear more then once in an OPSF FAD TLV. If it appears more then once, the OSPF FAD TLV MUST be ignored by the receiver.

7.3. OSPF Flexible Algorithm Include-All Admin Group Sub-TLV

The usage of this Sub-TLVs is described in Section 6.3.

The format of the OSPF Flexible Algorithm Include-Any Admin Group Sub-TLV is identical to the format of the OSPF FAEAG Sub-TLV in Section 7.1.

Flexible Algorithm Include-Any Admin Group Sub-TLV Type is 3.

OSPF Flexible Algorithm Include-All Admin Group Sub-TLV MAY NOT appear more then once in an OPSF FAD TLV. If it appears more then once, the OSPF FAD TLV MUST be ignored by the receiver.

8. Advertisement of Node Participation in a Flex-Algorithm

When a router is configured to support a particular Flex-Algorithm, we say it is participating in that Flex-Algorithm.

Paths computed for a specific Flex-Algorithm MAY be used by various applications, each potentially using its own specific data plane for forwarding the data over such paths. To guarantee the presence of the application specific forwarding state associated with a particular Flex-Algorithm, a router MUST advertise its participation for a particular Flex-Algorithm for each application specifically.

8.1. Advertisement of Node Participation for Segment Routing

[I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions] (IGP Segment Routing extensions) describe how SR-Algorithm is used to define how the best path is computed by the IGP.

Routers advertise the support for the SR-Algorithm as a node capability as described in the above mentioned IGP Segment Routing extensions. To advertise participation for a particular Flex-Algorithm for Segment Routing, the Flex-Algorithm value MUST be advertised in the SR-Algorithm TLV (OSPF) or sub-TLV (ISIS).

Segment Routing Flex-Algorithm participation advertisement is topology independent. When a router advertises participation in an SR-Algorithm, the participation applies to all topologies in which the advertising node participates.

8.2. Advertisement of Node Participation for Other Applications

This section describes considerations related to how other applications can advertise its participation in a specific Flex-Algorithm.

Application specific Flex-Algorithm participation advertisements MAY be topology specific or MAY be topology independent, depending on the application itself.

Application specific advertisement for Flex-Algorithm participation MUST be defined for each application and is outside of the scope of this document.

9. Advertisement of Link Attributes for Flex-Algorithm

Various link include or exclude rules can be part of the Flex-Algorithm definition. These rules use Admin Groups (AG) as defined in [RFC7308] and [RFC5305], or Extended Administrative Groups (EAG) as defined in [RFC7308].

To advertise a link affinity in a form of the AG or EAG that is used during Flex-Algorithm calculation, an Application Specific Link Attributes sub-TLV as described in [I-D.ietf-isis-te-app], or sub-TLV of Extended Link TLV as described in [I-D.ietf-ospf-te-link-attr-reuse] MUST be used. The advertisement MUST indicate that it is usable by the Flex-Algorithm application.

10. Calculation of Flexible Algorithm Paths

A router MUST be configured to participate in a given Flex-Algorithm K before it can compute any path for that Flex-Algorithm.

A router which participates in a given Flex Algorithm MUST use the FAD selected based on the rules defined in Section Section 5.3.

As described in Section 8, participation for any particular Flex-Algorithm MUST be advertised on a per application basis. Calculation of the paths for any particular Flex-Algorithm MUST be application specific.

The way applications handle nodes that do not participate in Flexible-Algorithm is application specific. If the application only wants to consider participating nodes during the Flex-Algorithm calculation, then when computing paths for a given Flex-Algorithm, all nodes that do not advertise participation for that Flex-Algorithm in the application specific advertisements MUST be pruned from the topology. MPLS Segment Routing is an application that MUST use such pruning when computing Flex-Algorithm paths.

When computing the path for a give Flex-Algorithm, the metric-type that is part of the Flex-Algorithm definition (Section 5) MUST be used.

When computing the path for a given Flex-Algorithm, the calculation-type that is part of the Flex-Algorithm definition (Section 5) MUST be used.

Various link include or exclude rules can be part of the Flex-Algorithm definition. To refer to particular bit within an AG or EAG we uses term 'color'.

Rules, in the order as specified below, MUST be used to prune link from the topology during the Flex-Algorithm computation.

For all links in the topology:

1. Check if any exclude rule is part of the Flex-Algorithm definition. If such exclude rule exists, check if any color that is part of the exclude rule is also set on the link. If such a color exist, the link MUST be pruned from the computation.

2. Check if any include-any rule is part of the Flex-Algorithm definition. if such include-any rule exists, check if any color that is part of the include-any rule is also set on the link. If such color does not exist, the link MUST be pruned from the computation.

3. Check if any include-all rule is part of the Flex-Algorithm definition. If such include-all rule exists, check if all colors that are part of the include-all rule are also set on the link. If not all such colors are set on the link, the link MUST be pruned from the computation.

4. If the Flex-Algorithm definition uses other than IGP metric (Section 5), and such metric is not advertised for the particular link in a topology for which the computation is done, such link MUST be pruned from the computation. A metric of value 0 MUST NOT be assumed in such case.

Any IGP Shortest Path Tree calculation is limited to a single area. Same applies to Flex-Algorithm calculations. Given that the computing router may not have the visibility to the topology of remote areas, the Flex-Algorithm specific path to an inter-area prefix will only be computed for the local area only. The egress L1/L2 router (ABR in OSPF) will be selected based on the best path for the given Flex-Algorithm in the local area and such egress L1/L2 (ABR in OSPF) router will be responsible to compute the best Flex-Algorithm specific path over the next area. This may produce an end-to-end path, which is sub-optimal based on Flex-Algorithm constraints. If the best end-to-end path for a given Flex-Algorithm needs to be used for inter-area destinations, paths for such

destinations need to be computed by the entity that has the topological information about all areas.

11. Flex-Algorithm and Forwarding Plane

This section describes how Flex-Algorithm paths are used with forwarding.

11.1. Segment Routing MPLS Forwarding for Flex-Algorithm

This section describes how Flex-Algorithm paths are used with SR MPLS forwarding.

Prefix SID advertisements include an SR-Algorithm value and as such are associated with the specified SR-Algorithm. Prefix-SIDs are also associated with a specific topology which is inherited from the associated prefix reachability advertisement. When the algorithm value advertised is a Flex-Algorithm value, the Prefix SID is associated with paths calculated using that Flex-Algorithm in the associated topology.

A Flex-Algorithm path MUST be installed in the MPLS forwarding plane using the MPLS label that corresponds to the Prefix-SID that was advertised for that Flex-algorithm. If the Prefix SID for a given Flex-algorithm is not known, the Flex-Algorithm specific path cannot be installed in the MPLS forwarding plane.

Traffic that is supposed to be routed via Flex-Algorithm specific paths, MUST be dropped where there are no such paths available.

Loop Free Alternate (LFA) paths for a given Flex-Algorithm MUST be computed using the same constraints as the calculation of the primary paths for that Flex-Algorithm. LFA paths MUST only use Prefix-SIDs advertised specifically for the given algorithm. LFA paths MUST NOT use an Adjacency-SID that belongs to a link that has been pruned from the Flex-Algorithm computation.

If LFA protection is being used to protect a given Flex-Algorithm paths, all routers in the area participating in the given Flex-Algorithm SHOULD advertise at least one Flex-Algorithm specific Node-SID. These Node-SIDs are used to enforce traffic over the LFA computed backup path.

11.2. Other Applications' Forwarding for Flex-Algorithm

Any application that wants to use Flex-Algorithm specific forwarding need to install some form of Flex-Algorithm specific forwarding entries.

Application specific forwarding for Flex-Algorithm MUST be defined for each application and is outside of the scope of this document.

12. Backward Compatibility

This extension brings no new backward compatibility issues.

13. Security Considerations

This draft adds a two new ways to disrupt the IGP networks:

An attacker can hijack a particular Flex-Algorithm by advertising a FAD with a priority of 255 (or any priority higher than that of the legitimate nodes).

An attacker could make it look like a router supports a particular Flex-Algorithm when it actually doesn’t, or vice versa.

Both of these attacks can be addressed by the existing security extensions as described in [RFC5304] and [RFC5310] for ISIS, in [RFC2328] and [RFC7474] for OSPFv2 and in [RFC5340] and [RFC4552] for OSPFv3.

14. IANA Considerations

14.1. IGP IANA Considerations

14.1.1. IGP Algorithm Types Registry

This document makes the following registrations in the "IGP Algorithm Types" registry:

Type: 128-255.

Description: Flexible Algorithms.

Reference: This document (Section 4).

14.1.2. Flexible Algorithm Definition Metric-Type Registry

IANA is requested to set up a registry called "Flexible Algorithm Definition Metric-Type Registry" under a "Interior Gateway Protocol (IGP) Parameters" IANA registries. The registration policy for this registry is "Standards Action" ([RFC8126] and [RFC7120]).

Values in this registry come from the range 0-255.

This document registers following values in the "Flexible Algorithm Definition Metric-Type Registry":

Type: 0

Description: IGP metric

Reference: This document (Section 5.1)

Type: 1

Description: Min Unidirectional Link Delay [RFC7810]

Reference: This document (Section 5.1)

Type: 2

Description: TE Default Metric [RFC5305]

Reference: This document (Section 5.1)

14.2. ISIS IANA Considerations

14.2.1. Sub TLVs for Type 242

This document makes the following registrations in the "sub-TLVs for TLV 242" registry.

Type: TBD (suggested value 26).

Description: Flexible Algorithm Definition Sub-TLV.

Reference: This document (Section 5.1).

14.2.2. Sub-Sub-TLVs for Flexible Algorithm Definition Sub-TLV

This document creates the following Sub-Sub-TLV Registry:

Registry: Sub-Sub-TLVs for Flexible Algorithm Definition Sub-TLV

Registration Procedure: Expert review

Reference: This document (Section 5.1)

This document defines the following Sub-Sub-TLVs in the "Sub-Sub-TLVs for Flexible Algorithm Definition Sub-TLV" registry:

Type: 1

Description: Flexible Algorithm Exclude Admin Group Sub-TLV

Reference: This document (Section 6.1).

Type: 2

Description: Flexible Algorithm Include-Any Admin Group Sub-TLV

Reference: This document (Section 6.2).

Type: 3

Description: Flexible Algorithm Include-All Admin Group Sub-TLV

Reference: This document (Section 6.3).

14.3. OSPF IANA Considerations

14.3.1. OSPF Router Information (RI) TLVs Registry

This specification updates the OSPF Router Information (RI) TLVs Registry with the following value:

o TBD (suggested value 16) - Flexible Algorithm Definition TLV

14.3.2. OSPF Flexible Algorithm Definition TLV Sub-TLV Registry

This document creates the following registry:

Registry: OSPF Flexible Algorithm Definition TLV sub-TLV

Registration Procedure: Expert review

Reference: This document (Section 5.2)

The "OSPF Flexible Algorithm Definition TLV sub-TLV" registry will define sub-TLVs at any level of nesting for Flexible Algorithm TLV and should be added to the "Open Shortest Path First (OSPF) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

This document resisters following Sub-TLVs in the "TLVs for Flexible Algorithm Definition TLV" registry:

Type: 1

Description: Flexible Algorithm Exclude Admin Group Sub-TLV

Reference: This document (Section 7.1).

Type: 2

Description: Flexible Algorithm Include-Any Admin Group Sub-TLV

Reference: This document (Section 7.2).

Type: 3

Description: Flexible Algorithm Include-All Admin Group Sub-TLV

Reference: This document (Section 7.3).

Types in the range 32768-33023 are for experimental use; these will not be registered with IANA, and MUST NOT be mentioned by RFCs.

Types in the range 33024-65535 are not to be assigned at this time. Before any assignments can be made in the 33024-65535 range, there MUST be an IETF specification that specifies IANA Considerations that covers the range being assigned.

15. Contributors

This draft, among other things, is also addressing the problem that the [I-D.gulkohegde-routing-planes-using-sr] was trying to solve. All authors of that draft agreed to join this draft.

Thanks to Eric Rosen, Les Ginsberg and Tony Przygienda for their detailed review and excellent comments.

Thanks to Cengiz Halit for his review and feedback during initial phase of the solution definition.

Thanks to Kenji Kumaki for his comments.

16. References

16.1. Normative References

[BCP14] , <https://tools.ietf.org/html/bcp14>.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-16 (work in progress), April 2018.

[I-D.ietf-isis-te-app]
Ginsberg, L., Psenak, P., Previdi, S., Henderickx, W., and J. Drake, "IS-IS TE Attributes per application", draft-ietf-isis-te-app-04 (work in progress), April 2018.

[I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Filsfils, C., Previdi, S., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-12 (work in progress), April 2018.

[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-25 (work in progress), April 2018.

[I-D.ietf-ospf-te-link-attr-reuse]
Psenak, P., Lindem, A., Ginsberg, L., Henderickx, W., Tantsura, J., Gredler, H., and J. Drake, "OSPFv2 Link Traffic Engineering (TE) Attribute Reuse", draft-ietf-ospf-te-link-attr-reuse-03 (work in progress), January 2018.

[ISO10589]
International Organization for Standardization, "Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Second Edition, Nov 2002.

[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>.

[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>.

[RFC7770] Lindem, A., Ed., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 7770, DOI 10.17487/RFC7770, February 2016, <https://www.rfc-editor.org/info/rfc7770>.

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <https://www.rfc-editor.org/info/rfc7981>.

[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>.

16.2. Informative References

[I-D.gulkohegde-routing-planes-using-sr] Hegde, S. and a. arkadiy.gulko@thomsonreuters.com, "Separating Routing Planes using Segment Routing", draft-gulkohegde-routing-planes-using-sr-00 (work in progress), March 2017.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3906] Shen, N. and H. Smit, "Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels", RFC 3906, DOI 10.17487/RFC3906, October 2004, <https://www.rfc-editor.org/info/rfc3906>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <https://www.rfc-editor.org/info/rfc4552>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[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>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <https://www.rfc-editor.org/info/rfc7474>.

[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>.

[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>.

Authors' Addresses

Peter Psenak (editor)
Cisco Systems
Apollo Business Center
Mlynske nivy 43
Bratislava, 82109
Slovakia

Email: ppsenak@cisco.com

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

Email: shraddha@juniper.net

Clarence Filsfils
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.
S.No. 154/6, Phase I, Hinjawadi
PUNE, MAHARASHTRA 411 057
India

Email: ketant@cisco.com

Arkadiy Gulko
Thomson Reuters

Email: arkadiy.gulko@thomsonreuters.com

Internet
Internet-Draft
Intended status: Informational
Expires: January 2, 2019

D. Yeung
Arrcus
Y. Qu
Huawei
J. Zhang
Juniper Networks
I. Chen
Jabil Circuit
A. Lindem
Cisco Systems
July 1, 2018

Yang Data Model for OSPF SR (Segment Routing) Protocol
draft-ietf-ospf-sr-yang-05

Abstract

This document defines a YANG data model that can be used to configure and manage OSPF Segment Routing.

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 2, 2019.

Copyright Notice

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

YANG [RFC6020] [RFC7950] is a data definition language used to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g., ReST) and encodings other than XML (e.g., JSON) are being defined. Furthermore, YANG data models can be used as the basis for implementation of other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage OSPF Segment Routing and it is an augmentation to the OSPF YANG data model.

1.1. 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].

2. OSPF Segment Routing

This document defines a model for OSPF Segment Routing feature [I-D.ietf-ospf-segment-routing-extensions][I-D.ietf-ospf-ospfv3-segment-routing-extensions]. It is an augmentation of the OSPF base model.

The OSPF SR YANG module requires support for the base segment routing module [I-D.ietf-spring-sr-yang], which defines the global segment

routing configuration independent of any specific routing protocol configuration, and support of OSPF base model[I-D.ietf-ospf-yang] which defines basic OSPF configuration and state.

```
module: ietf-ospf-sr
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol/ospf:ospf:
    +--rw segment-routing
    |  +--rw enabled?    boolean
    |  +--rw bindings
    |     +--rw advertise
    |     |  +--rw policies*   string
    |     +--rw receive?     boolean
    +--rw protocol-srgb {sr:protocol-srgb}?
       +--rw srgb* [lower-bound upper-bound]
          +--rw lower-bound    uint32
          +--rw upper-bound    uint32
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol/ospf:ospf
          /ospf:areas/ospf:area/ospf:interface:
    +--rw segment-routing
       +--rw adjacency-sid
          +--rw advertise-adj-group-sid* [group-id]
          |  +--rw group-id    uint32
          +--rw advertise-protection?      enumeration
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol/ospf:ospf
          /ospf:areas/ospf:area/ospf:interfaces/ospf:interface
          /ospf:fast-reroute:
    +--rw ti-lfa {ti-lfa}?
       +--rw enable?   boolean
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol/ospf:ospf
          /ospf:areas/ospf:area/ospf:interfaces/ospf:interface
          /ospf:database/ospf:link-scope-lsa-type/ospf:link-scope-lsas
          /ospf:link-scope-lsa/ospf:version/ospf:ospfv2/ospf:ospfv2
          /ospf:body/ospf:opaque/ospf:extended-prefix-tlvs
          /ospf:extended-prefix-tlv:
    +--ro perfix-sid-sub-tlvs
       +--ro prefix-sid-sub-tlv*
          +--ro flags?       bits
          +--ro mt-id?       uint8
          +--ro algorithm?   uint8
          +--ro sid?         uint32
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol/ospf:ospf
          /ospf:areas/ospf:area/ospf:database/ospf:area-scope-lsa-type
          /ospf:area-scope-lsas/ospf:area-scope-lsa/ospf:version
```

```
           /ospf:ospfv2/ospf:ospfv2/ospf:body/ospf:opaque
           /ospf:extended-prefix-tlvs/ospf:extended-prefix-tlv:
     +--ro perfix-sid-sub-tlvs
        +--ro prefix-sid-sub-tlv*
           +--ro flags?       bits
           +--ro mt-id?       uint8
           +--ro algorithm?   uint8
           +--ro sid?         uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/ospf:ospf
           /ospf:database/ospf:as-scope-lsa-type/ospf:as-scope-lsas
           /ospf:as-scope-lsa/ospf:version/ospf:ospfv2/ospf:ospfv2
           /ospf:body/ospf:opaque/ospf:extended-prefix-tlvs
           /ospf:extended-prefix-tlv:
     +--ro perfix-sid-sub-tlvs
        +--ro prefix-sid-sub-tlv*
           +--ro flags?       bits
           +--ro mt-id?       uint8
           +--ro algorithm?   uint8
           +--ro sid?         uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/ospf:ospf
           /ospf:areas/ospf:area/ospf:database/ospf:area-scope-lsa-type
           /ospf:area-scope-lsas/ospf:area-scope-lsa/ospf:version
           /ospf:ospfv2/ospf:ospfv2/ospf:body/ospf:opaque
           /ospf:extended-link-tlvs/ospf:extended-link-tlv:
     +--ro adj-sid-sub-tlvs
     |  +--ro adj-sid-sub-tlv*
     |     +--ro flags?    bits
     |     +--ro mt-id?    uint8
     |     +--ro weight?   uint8
     |     +--ro sid?      uint32
     +--ro lan-adj-sid-sub-tlvs
        +--ro lan-adj-sid-sub-tlv*
           +--ro flags?                bits
           +--ro mt-id?                uint8
           +--ro weight?               uint8
           +--ro neighbor-router-id?   yang:dotted-quad
           +--ro sid?                  uint32
   augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol/ospf:ospf
           /ospf:areas/ospf:area/ospf:interfaces/ospf:interface
           /ospf:database/ospf:link-scope-lsa-type/ospf:link-scope-lsas
           /ospf:link-scope-lsa/ospf:version/ospf:ospfv2/ospf:ospfv2
           /ospf:body/ospf:opaque:
     +--ro extended-prefix-range-tlvs
     |  +--ro extended-prefix-range-tlv*
     |     +--ro range-size?            uint16
```

```
      |     +--ro flags?                bits
      |     +--ro prefix?               inet:ip-prefix
      |     +--ro perfix-sid-sub-tlvs
      |     |  +--ro prefix-sid-sub-tlv*
      |     |     +--ro flags?       bits
      |     |     +--ro mt-id?       uint8
      |     |     +--ro algorithm?   uint8
      |     |     +--ro sid?         uint32
      |     +--ro unknown-tlvs
      |        +--ro unknown-tlv*
      |           +--ro type?     uint16
      |           +--ro length?   uint16
      |           +--ro value?    yang:hex-string
      +--ro sr-algorithm-tlv
      |  +--ro sr-algorithm*   uint8
      +--ro sid-range-tlvs
      |  +--ro sid-range-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro local-block-tlvs
      |  +--ro local-block-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro srms-preference-tlv
         +--ro preference?   uint8
    augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol/ospf:ospf
            /ospf:areas/ospf:area/ospf:database/ospf:area-scope-lsa-type
            /ospf:area-scope-lsas/ospf:area-scope-lsa/ospf:version
            /ospf:ospfv2/ospf:ospfv2/ospf:body/ospf:opaque:
      +--ro extended-prefix-range-tlvs
      |  +--ro extended-prefix-range-tlv*
      |     +--ro range-size?            uint16
      |     +--ro flags?                 bits
      |     +--ro prefix?                inet:ip-prefix
      |     +--ro perfix-sid-sub-tlvs
      |     |  +--ro prefix-sid-sub-tlv*
      |     |     +--ro flags?       bits
      |     |     +--ro mt-id?       uint8
      |     |     +--ro algorithm?   uint8
      |     |     +--ro sid?         uint32
      |     +--ro unknown-tlvs
      |        +--ro unknown-tlv*
      |           +--ro type?     uint16
      |           +--ro length?   uint16
      |           +--ro value?    yang:hex-string
```

```
      +--ro sr-algorithm-tlv
      |  +--ro sr-algorithm*   uint8
      +--ro sid-range-tlvs
      |  +--ro sid-range-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro local-block-tlvs
      |  +--ro local-block-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro srms-preference-tlv
         +--ro preference?   uint8
    augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol/ospf:ospf
            /ospf:database/ospf:as-scope-lsa-type/ospf:as-scope-lsas
            /ospf:as-scope-lsa/ospf:version/ospf:ospfv2/ospf:ospfv2
            /ospf:body/ospf:opaque:
      +--ro extended-prefix-range-tlvs
      |  +--ro extended-prefix-range-tlv*
      |     +--ro range-size?            uint16
      |     +--ro flags?                 bits
      |     +--ro prefix?                inet:ip-prefix
      |     +--ro perfix-sid-sub-tlvs
      |     |  +--ro prefix-sid-sub-tlv*
      |     |     +--ro flags?       bits
      |     |     +--ro mt-id?       uint8
      |     |     +--ro algorithm?   uint8
      |     |     +--ro sid?         uint32
      |     +--ro unknown-tlvs
      |        +--ro unknown-tlv*
      |           +--ro type?     uint16
      |           +--ro length?   uint16
      |           +--ro value?    yang:hex-string
      +--ro sr-algorithm-tlv
      |  +--ro sr-algorithm*   uint8
      +--ro sid-range-tlvs
      |  +--ro sid-range-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro local-block-tlvs
      |  +--ro local-block-tlv*
      |     +--ro range-size?    ospf:uint24
      |     +--ro sid-sub-tlv
      |        +--ro sid?   uint32
      +--ro srms-preference-tlv
```

```
         +--ro preference?   uint8
```

3. OSPF Segment Routing Yang Module

```
   <CODE BEGINS> file "ietf-ospf-sr@2018-06-25.yang"
   module ietf-ospf-sr {
     namespace "urn:ietf:params:xml:ns:yang:ietf-ospf-sr";

     prefix ospf-sr;

     import ietf-inet-types {
       prefix "inet";
     }

     import ietf-yang-types {
       prefix "yang";
     }

     import ietf-routing {
       prefix "rt";
     }
     import ietf-segment-routing-common {
       prefix "sr-cmn";
     }
     import ietf-segment-routing {
       prefix "sr";
     }
     import ietf-ospf {
       prefix "ospf";
     }

     organization
       "IETF OSPF - OSPF Working Group";

     contact
        "WG Web:   <http://tools.ietf.org/wg/ospf/>
         WG List:  <mailto:ospf@ietf.org>

         Editor:   Derek Yeung
                   <mailto:derek@arrcus.com>
         Author:   Derek Yeung
                   <mailto:derek@arrcus.com>
         Author:   Yingzhen Qu
                   <mailto:yingzhen.qu@huawei.com>
         Author:   Acee Lindem
                   <mailto:acee@cisco.com>
         Author:   Jeffrey Zhang
                   <mailto:zzhang@juniper.net>
```

```
         Author:   Ing-Wher Chen
                   <mailto:ing-wher_chen@jabil.com>
         Author:   Greg Hankins
                   <mailto:greg.hankins@alcatel-lucent.com>";

     description
       "This YANG module defines the generic configuration
        and operational state for OSPF Segment Routing, which is
        common across all of the vendor implementations. It is
        intended that the module will be extended by vendors to
        define vendor-specific OSPF Segment Routing configuration
        and operational parameters and policies.

        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).

        This version of this YANG module is part of RFC XXXX;
        see the RFC itself for full legal notices.";

     reference "RFC XXXX";

     revision 2018-06-25 {
       description
         "";
       reference
         "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
     }

     revision 2018-03-03 {
       description
         "* Remove OSPF instance.";
       reference
         "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
     }

     revision 2017-12-28 {
       description
         "";
       reference
         "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
     }
```

```
    revision 2017-07-02 {
      description
        "* Implement NMDA model.
         * Add local-block-tlvs and srms-preference-tlv.
         * Remove sid-binding-sub-tlvs.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2017-03-12 {
      description
        "* Add p-flag in adj-sid sub-tlv.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }
    revision 2016-10-31 {
      description
        "* Update authors information.
         * Add import of ietf-segment-routing-common module.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2016-07-07 {
      description
        "* Change routing-protocol to control-plane-protocol.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2016-03-20 {
      description
        "* Remove routing-instance.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2015-10-19 {
      description
        "* Add per-protocol SRGB support.
         * Editorial changes.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2015-09-02 {
      description
        "* Author list update.
```

```
          * Editorial changes.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    revision 2015-07-06 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for OSPF Segment Routing.";
    }

    feature ti-lfa {
      description
        "Topology-Independent Loop-Free Alternate (TI-LFA)
         computation using segment routing.";
    }

    /* Groupings */
    grouping sid-sub-tlv {
      description "SID/Label sub-TLV grouping.";
      container sid-sub-tlv {
        description
          "Used to advertise the SID/Label associated with a
           prefix or adjacency.";
        leaf sid {
          type uint32;
          description
            "Segment Identifier (SID) - A 20 bit label or
             32 bit SID.";
        }
      }
    }

    grouping prefix-sid-sub-tlvs {
      description "Prefix Segment ID (SID) sub-TLVs.";
      container perfix-sid-sub-tlvs{
        description "Prefix SID sub-TLV.";
        list prefix-sid-sub-tlv {
          description "Prefix SID sub-TLV.";
          leaf flags {
            type bits {
              bit NP {
                position 1;
                description
                  "No-PHP flag.";
              }
              bit M {
```

```
                position 2;
                description
                  "Mapping server flag.";
              }
              bit E {
                position 3;
                description
                  "Explicit-NULL flag.";
              }
              bit V {
                position 4;
                description
                  "Value/Index flag.";
              }
              bit L {
                position 5;
                description
                  "Local flag.";
              }
            }
            description "Segment Identifier (SID) Flags.";
          }
          leaf mt-id {
            type uint8;
            description "Multi-topology ID.";
          }
          leaf algorithm {
            type uint8;
            description
              "The algorithm associated with the prefix-SID.";
          }
          leaf sid {
            type uint32;
            description "An index or label.";
          }
        }
      }
    }

    grouping extended-prefix-range-tlvs {
      description "Extended prefix range TLV grouping.";

      container extended-prefix-range-tlvs {
        description "The list of range of prefixes.";
        list extended-prefix-range-tlv {    //type=2?
          description "The range of prefixes.";
          leaf range-size {
            type uint16;
```

```
            description "The number of prefixes covered by the
                         advertisement.";
          }
          leaf flags {
            type bits {
              bit IA {
                position 0;
                description
                  "Inter-Area flag.";
              }
            }
            description "Flags.";
          }
          leaf prefix {
            type inet:ip-prefix;
            description "Address prefix.";
          }
          uses prefix-sid-sub-tlvs;
          uses ospf:unknown-tlvs;
        }
      }
    }

    grouping sr-algorithm-tlv {
      description "SR algorithm TLV grouping.";
      container sr-algorithm-tlv {
        description "All SR algorithm TLVs.";
        leaf-list sr-algorithm {
          type uint8;
          description
            "The Segment Routing (SR) algorithms that the router is
             currently using.";
        }
      }
    }

    grouping sid-range-tlvs {
      description "SID Range TLV grouping.";
      container sid-range-tlvs {
        description "List of SID range TLVs.";
        list sid-range-tlv {
          description "SID range TLV.";
          leaf range-size {
            type ospf:uint24;
            description "The SID range.";
          }
          uses sid-sub-tlv;
        }
```

```
      }
    }

    grouping local-block-tlvs {
      description "The SR local block TLV contains the
                   range of labels reserved for local SIDs.";
      container local-block-tlvs {
        description "List of SRLB TLVs.";
        list local-block-tlv {
          description "SRLB TLV.";
          leaf range-size {
            type ospf:uint24;
            description "The SID range.";
          }
          uses sid-sub-tlv;
        }
      }
    }

    grouping srms-preference-tlv {
      description "The SRMS preference TLV is used to advertise
                   a preference associated with the node that acts
                   as an SR Mapping Server.";
      container srms-preference-tlv {
        description "SRMS Preference TLV.";
        leaf preference {
          type uint8 {
            range "0 .. 255";
          }
          description "SRMS preference TLV, vlaue from 0 to 255.";
        }
      }
    }

    /* Configuration */
    augment "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/ospf:ospf" {
      when "../rt:type = 'ospf:ospfv2' or "
         + "../rt:type = 'ospf:ospfv3'" {
        description
          "This augments the OSPF routing protocol when used.";
      }
      description
        "This augments the OSPF protocol configuration
         with segment routing.";
      uses sr:controlplane-cfg;
      container protocol-srgb {
        if-feature sr:protocol-srgb;
```

```
        uses sr-cmn:srgb-cfg;
        description
          "Per-protocol SRGB.";
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/ospf:ospf/"
          + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface" {
      when "../../../../../rt:type = 'ospf:ospfv2' or "
         + "../../../../../rt:type = 'ospf:ospfv3'" {
        description
          "This augments the OSPF interface configuration
           when used.";
      }
      description
        "This augments the OSPF protocol interface
         configuration with segment routing.";

      uses sr:igp-interface-cfg;
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/ospf:ospf/"
          + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface/"
          + "ospf:fast-reroute" {
      when "../../../../../../rt:type = 'ospf:ospfv2' or "
         + "../../../../../../rt:type = 'ospf:ospfv3'" {
        description
          "This augments the OSPF routing protocol when used.";
      }
      description
        "This augments the OSPF protocol IP-FRR with TI-LFA.";

      container ti-lfa {
        if-feature ti-lfa;
        leaf enable {
          type boolean;
          description
           "Enables TI-LFA computation.";
        }
        description
          "Topology Independent Loop Free Alternate
           (TI-LFA) support.";
      }
    }

    /* Database */
```

```
    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:areas/ospf:area/"
          + "ospf:interfaces/ospf:interface/ospf:database/"
          + "ospf:link-scope-lsa-type/ospf:link-scope-lsas/"
          + "ospf:link-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque/"
          + "ospf:extended-prefix-tlvs/ospf:extended-prefix-tlv" {
      when "../../../../../../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }
      description
        "SR specific TLVs for OSPFv2 extended prefix TLV
         in type 9 opaque LSA.";
      uses prefix-sid-sub-tlvs;
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:areas/"
          + "ospf:area/ospf:database/"
          + "ospf:area-scope-lsa-type/ospf:area-scope-lsas/"
          + "ospf:area-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque/"
          + "ospf:extended-prefix-tlvs/ospf:extended-prefix-tlv" {
      when "../../../../../../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }
      description
        "SR specific TLVs for OSPFv2 extended prefix TLV
         in type 10 opaque LSA.";
      uses prefix-sid-sub-tlvs;
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:database/"
          + "ospf:as-scope-lsa-type/ospf:as-scope-lsas/"
          + "ospf:as-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque/"
          + "ospf:extended-prefix-tlvs/ospf:extended-prefix-tlv" {
      when "../../../../../../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
```

```
          "This augmentation is only valid for OSPFv2.";
      }
      description
        "SR specific TLVs for OSPFv2 extended prefix TLV
         in type 11 opaque LSA.";
      uses prefix-sid-sub-tlvs;
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:areas/"
          + "ospf:area/ospf:database/"
          + "ospf:area-scope-lsa-type/ospf:area-scope-lsas/"
          + "ospf:area-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque/"
          + "ospf:extended-link-tlvs/ospf:extended-link-tlv" {
      when "../../../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }
      description
        "SR specific TLVs for OSPFv2 extended link TLV
         in type 10 opaque LSA.";

      container adj-sid-sub-tlvs {
        description "Adjacency SID optional sub-TLVs.";
        list adj-sid-sub-tlv {
          description "List of Adjacency SID sub-TLVs.";
          leaf flags {
            type bits {
              bit B {
                position 0;
                description
                  "Backup flag.";
              }
              bit V {
                position 1;
                description
                  "Value/Index flag.";
              }
              bit L {
                position 2;
                description
                  "Local/Global flag.";
              }
              bit G {
                position 3;
```

```
                description
                  "Group flag.";
              }
              bit P {
                position 4;
                description
                  "Persistent flag.";
              }
            }
            description "Flags.";
          }
          leaf mt-id {
            type uint8;
            description "Multi-topology ID.";
          }
          leaf weight {
            type uint8;
            description "Weight used for load-balancing.";
          }
          leaf sid {
            type uint32;
            description "Segment Identifier (SID) index/label.";
          }
        }
      }

      container lan-adj-sid-sub-tlvs {
        description "LAN Adjacency SID optional sub-TLVs.";
        list lan-adj-sid-sub-tlv {
          description "List of LAN adjacency SID sub-TLVs.";
          leaf flags {
            type bits {
              bit B {
                position 0;
                description
                  "Backup flag.";
              }
              bit V {
                position 1;
                description
                  "Value/Index flag.";
              }
              bit L {
                position 2;
                description
                  "Local/Global flag.";
              }
              bit G {
```

```
                position 3;
                description
                  "Group flag.";
              }
              bit P {
                position 4;
                description
                  "Persistent flag.";
              }
            }
            description "Flags.";
          }
          leaf mt-id {
            type uint8;
            description "Multi-topology ID.";
          }
          leaf weight {
            type uint8;
            description "Weight used for load-balancing.";
          }
          leaf neighbor-router-id {
            type yang:dotted-quad;
            description "Neighbor router ID.";
          }
          leaf sid {
            type uint32;
            description "Segment Identifier (SID) index/label.";
          }
        }
      }
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:areas/ospf:area/"
          + "ospf:interfaces/ospf:interface/ospf:database/"
          + "ospf:link-scope-lsa-type/ospf:link-scope-lsas/"
          + "ospf:link-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque" {
      when "../../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }

      description
        "SR specific TLVs for OSPFv2 type 9 opaque LSA.";
```

```
      uses extended-prefix-range-tlvs;
      uses sr-algorithm-tlv;
      uses sid-range-tlvs;
      uses local-block-tlvs;
      uses srms-preference-tlv;
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:areas/"
          + "ospf:area/ospf:database/"
          + "ospf:area-scope-lsa-type/ospf:area-scope-lsas/"
          + "ospf:area-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque" {
      when "../../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }

      description
        "SR specific TLVs for OSPFv2 type 10 opaque LSA.";

      uses extended-prefix-range-tlvs;
      uses sr-algorithm-tlv;
      uses sid-range-tlvs;
      uses local-block-tlvs;
      uses srms-preference-tlv;
    }

    augment "/rt:routing/"
          + "rt:control-plane-protocols/rt:control-plane-protocol/"
          + "ospf:ospf/ospf:database/"
          + "ospf:as-scope-lsa-type/ospf:as-scope-lsas/"
          + "ospf:as-scope-lsa/ospf:version/ospf:ospfv2/"
          + "ospf:ospfv2/ospf:body/ospf:opaque" {
      when "../../../../../../../../../"
         + "rt:type = 'ospf:ospfv2'" {
        description
          "This augmentation is only valid for OSPFv2.";
      }
      description
        "SR specific TLVs for OSPFv2 type 11 opaque LSA.";

      uses extended-prefix-range-tlvs;
      uses sr-algorithm-tlv;
      uses sid-range-tlvs;
      uses local-block-tlvs;
```

```
      uses srms-preference-tlv;
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The data model defined does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

5. Acknowledgements

The authors wish to thank Yi Yang, Alexander Clemm, Gaurav Gupta, Ladislav Lhotka, Stephane Litkowski, Greg Hankins, Manish Gupta and Alan Davey for their thorough reviews and helpful comments.

This document was produced using Marshall Rose’s xml2rfc tool.

6. References

6.1. Normative References

[I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Filsfils, C., Previdi, S., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-13 (work in progress), May 2018.

[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-25 (work in progress), April 2018.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-11 (work in progress), April 2018.

[I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-09 (work in progress), June 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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC4750] Joyal, D., Ed., Galecki, P., Ed., Giacalone, S., Ed., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, DOI 10.17487/RFC4750, December 2006, <https://www.rfc-editor.org/info/rfc4750>.

[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>.

[RFC5643] Joyal, D., Ed. and V. Manral, Ed., "Management Information Base for OSPFv3", RFC 5643, DOI 10.17487/RFC5643, August 2009, <https://www.rfc-editor.org/info/rfc5643>.

[RFC5838] Lindem, A., Ed., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, DOI 10.17487/RFC5838, April 2010, <https://www.rfc-editor.org/info/rfc5838>.

[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>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[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>.

6.2. Informative References

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

Appendix A. Contributors' Addreses

Dean Bogdanovic
Volta Networks, Inc.

EMail: dean@voltanet.io

Kiran Koushik Agrahara Sreenivasa
Cisco Systems
12515 Research Blvd, Bldg 4
Austin, TX 78681
USA

EMail: kkoushik@cisco.com

Authors' Addresses

Derek Yeung
Arrcus

EMail: derek@arrcus.com

Yingzhen Qu
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: yingzhen.qu@huawei.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: zzhang@juniper.net

Ing-Wher Chen
Jabil Circuit

EMail: Ing-Wher_chen@jabil.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513

EMail: acee@cisco.com

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2018

P. Psenak, Ed.
Cisco Systems, Inc.
A. Lindem
L. Ginsberg
Cisco Systems
W. Henderickx
Nokia
J. Tantsura
Nuage Networks
H. Gredler
RtBrick Inc.
J. Drake
Juniper Networks
June 15, 2018

OSPF Link Traffic Engineering (TE) Attribute Reuse
draft-ietf-ospf-te-link-attr-reuse-04.txt

Abstract

Various link attributes have been defined in OSPF in the context of the MPLS Traffic Engineering (TE) and GMPLS. Many of these link attributes can be used for applications other than MPLS Traffic Engineering or GMPLS. This document defines how to distribute such attributes in OSPFv2 and OSPFv3 for applications other than MPLS Traffic Engineering or GMPLS.

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 17, 2018.

Table of Contents

1. Introduction

Various link attributes have been defined in OSPFv2 [RFC2328] and OSPFv3 [RFC5340] in the context of the MPLS traffic engineering and GMPLS. All these attributes are distributed by OSPFv2 as sub-TLVs of the Link-TLV advertised in the OSPFv2 TE Opaque LSA [RFC3630]. In OSPFv3, they are distributed as sub-TLVs of the Link-TLV advertised in the OSPFv3 Intra-Area-TE-LSA as defined in [RFC5329].

Many of these link attributes are useful outside of traditional MPLS Traffic Engineering or GMPLS. This brings its own set of problems, in particular how to distribute these link attributes in OSPFv2 and OSPFv3 when MPLS TE and GMPLS are not deployed or are deployed in parallel with other applications that use these link attributes.

[RFC7855] discusses use cases/requirements for SR. Included among these use cases is SRTE. If both RSVP-TE and SRTE are deployed in a network, link attribute advertisements can be used by one or both of these applications. As there is no requirement for the link attributes advertised on a given link used by SRTE to be identical to the link attributes advertised on that same link used by RSVP-TE, there is a clear requirement to indicate independently which link attribute advertisements are to be used by each application.

As the number of applications which may wish to utilize link attributes may grow in the future, an additional requirement is that the extensions defined allow the association of additional applications to link attributes without altering the format of the advertisements or introducing new backwards compatibility issues.

Finally, there may still be many cases where a single attribute value can be shared among multiple applications, so the solution should minimize advertising duplicate link/attribute when possible.

1.1. Requirements notation

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. Link attributes examples

This section lists some of the link attributes originally defined for MPLS Traffic Engineering that can be used for other applications in OSPFv2 and OSPFv3. The list doesn't necessarily contain all the required attributes.

1. Remote Interface IP address [RFC3630] - OSPFv2 currently cannot distinguish between parallel links between two OSPFv2 routers. As a result, the two-way connectivity check performed during SPF may succeed when the two routers disagree on which of the links to use for data traffic.

2. Link Local/Remote Identifiers - [RFC4203] - Used for the two-way connectivity check for parallel unnumbered links. Also used for identifying adjacencies for unnumbered links in Segment Routing traffic engineering.

3. Shared Risk Link Group (SRLG) [RFC4203] - In IPFRR, the SRLG is used to compute diverse backup paths [RFC5714].

4. Unidirectional Link Delay/Loss Metrics [RFC7471] - Could be used for the shortest path first (SPF) computation using alternate metrics within an OSPF area.

3. Advertising Link Attributes

This section outlines possible approaches for advertising link attributes originally defined for MPLS Traffic Engineering or GMPLS when they are used for other applications.

3.1. OSPFv2 TE Opaque LSA and OSPFv3 Intra-Area-TE-LSA

One approach for advertising link attributes is to continue to use the OSPFv2 TE Opaque LSA [RFC3630] or the OSPFv3 Intra-Area-TE-LSA [RFC5329]. There are several problems with this approach:

1. Whenever the link is advertised in an OSPFv2 TE Opaque LSA or in an OSPFv3 Intra-Area-TE-LSA, the link becomes a part of the TE topology, which may not match IP routed topology. By making the link part of the TE topology, remote nodes may mistakenly believe that the link is available for MPLS TE or GMPLS, when, in fact, MPLS is not enabled on the link.

2. The OSPFv2 TE Opaque LSA and OSPFv3 Intra-Area-TE-LSA advertise link attributes that are not used or required by MPLS TE or GMPLS. There is no mechanism in these TE LSAs to indicate which

of the link attributes are passed to the MPLS TE application and which are used by other applications including OSPF itself.

3. Link attributes used for non-TE applications are partitioned across multiple LSAs - the TE Opaque LSA and the Extended Link Opaque LSA in OSPFv2 and the OSPFv3 Intra-Area-TE-LSA and OSPFv3 Extended LSA Router-Link TLV [RFC8362] in OSPFv3. This partitioning will require implementations to lookup multiple LSAs to extract link attributes for a single link, bringing needless complexity to OSPF implementations.

The advantage of this approach is that there is no additional standardization requirement to advertise the TE/GMPL attributes for other applications. Additionally, link attributes are only advertised once when both OSPF TE and other applications are deployed on the same link. This is not expected to be a common deployment scenario.

3.2. OSPFv2 Extended Link Opaque LSA and OSPFv3 E-Router-LSA

An alternative approach for advertising link attributes is to use Extended Link Opaque LSAs as defined in [RFC7684] for OSPFv2 and Extended Router-LSAs [RFC8362] for OSPFv3. These LSAs were defined as a generic containers for distribution of the extended link attributes. There are several advantages in using them:

1. Advertisement of the link attributes does not make the link part of the TE topology. It avoids any conflicts and is fully compatible with the [RFC3630] and [RFC5329].

2. The OSPFv2 TE Opaque LSA and OSPFv3 Intra-Area-TE-LSA remains truly opaque to OSPFv2 and OSPFv3 as originally defined in [RFC3630] and [RFC5329] respectively. Their contents are not inspected by OSPF, that act as a pure transport.

3. There is clear distinction between link attributes used by TE and link attributes used by other OSPFv2 or OSPFv3 applications.

4. All link attributes that are used by other applications are advertised in a single LSA, the Extended Link Opaque LSA in OSPFv2 or the OSPFv3 E-Router-LSA [RFC8362] in OSPFv3.

The disadvantage of this approach is that in rare cases, the same link attribute is advertised in both the TE Opaque and Extended Link Attribute LSAs in OSPFv2 or the Intra-Area-TE-LSA and E-Router-LSA in OSPFv3. Additionally, there will be additional standardization effort. However, this could also be viewed as an advantage as the

non-TE use cases for the TE link attributes are documented and validated by the LSR working group.

3.3. Selected Approach

It is RECOMMENDED to use the Extended Link Opaque LSA [RFC7684] and E-Router-LSA [RFC8362] to advertise any link attributes used for non-TE applications in OSPFv2 or OSPFv3 respectively, including those that have been originally defined for TE applications.

It is also RECOMMENDED that TE link attributes used for RSVP-TE/GMPLS continue to use OSPFv2 TE Opaque LSA [RFC3630] and OSPFv3 Intra-Area-TE-LSA [RFC5329].

It is also RECOMMENDED to keep the format of the link attribute TLVs that have been defined for TE applications unchanged even when they are used for non-TE applications.

Finally, it is RECOMMENDED to allocate unique code points for these TE link attribute TLVs in the OSPFv2 Extended Link TLV Sub-TLV Registry [RFC7684] and in the OSPFv3 Extended LSA Sub-TLV Registry [RFC8362]. For each reused TLV, the code point will be defined in an IETF document along with the expected use-case(s).

4. Reused TE link attributes

This section defines the use case and code points for the OSPFv2 Extended Link TLV Sub-TLV Registry and OSPFv3 Extended LSA Sub-TLV Registry for some of the link attributes that have been originally defined for TE or GMPLS.

Remote interface IP address and Link Local/Remote Identifiers have been added as sub-TLVs of OSPFv2 Extended Link TLV by [RFC8379]. Link Local/Remote Identifiers are already included in the OSPFv3 Router-Link TLV [RFC8362].

4.1. Shared Risk Link Group (SRLG)

The SRLG of a link can be used in IPFRR to compute a backup path that does not share any SRLG group with the protected link.

To advertise the SRLG of the link in the OSPFv2 Extended Link TLV, the same format for the sub-TLV defined in section 1.3 of [RFC4203] is used and TLV type TBD1 is used. Similarly, for OSPFv3 to advertise the SRLG in the OSPFv3 Router-Link TLV, TLV type TBD2 is used.

4.2. Extended Metrics

[RFC3630] defines several link bandwidth types. [RFC7471] defines extended link metrics that are based on link bandwidth, delay and loss characteristics. All these can be used to compute best paths within an OSPF area to satisfy requirements for bandwidth, delay (nominal or worst case) or loss.

To advertise extended link metrics in the OSPFv2 Extended Link TLV, the same format for the sub-TLVs defined in [RFC7471] is used with the following TLV types:

TBD3 - Unidirectional Link Delay

TBD4 - Min/Max Unidirectional Link Delay

TBD5 - Unidirectional Delay Variation

TBD6 - Unidirectional Link Loss

TBD7 - Unidirectional Residual Bandwidth

TBD8 - Unidirectional Available Bandwidth

TBD9 - Unidirectional Utilized Bandwidth

To advertise extended link metrics in the OSPFv3 Extended LSA Router-Link TLV, the same format for the sub-TLVs defined in [RFC7471] is used with the following TLV types:

TBD10 - Unidirectional Link Delay

TBD11 - Min/Max Unidirectional Link Delay

TBD12 - Unidirectional Delay Variation

TBD13 - Unidirectional Link Loss

TBD14 - Unidirectional Residual Bandwidth

TBD15 - Unidirectional Available Bandwidth

TBD16 - Unidirectional Utilized Bandwidth

4.3. Administrative Group

[RFC3630] and [RFC7308] define the Administrative Group and Extended Administrative Group sub-TLVs respectively.

One use case where advertisement of the Extended Administrative Group(s) for a link is required is described in [I-D.ietf-lsr-flex-algo].

To advertise the Administrative Group and Extended Administrative Group in the OSPFv2 Extended Link TLV, the same format for the sub-TLVs defined in [RFC3630] and [RFC7308] is used with the following TLV types:

TBD17 - Administrative Group

TBD18 - Extended Administrative Group

To advertise Administrative Group and Extended Administrative Group in the OSPFv3 Router-Link TLV, the same format for the sub-TLVs defined in [RFC3630] and [RFC7308] is used with the following TLV types:

TBD19 - Administrative Group

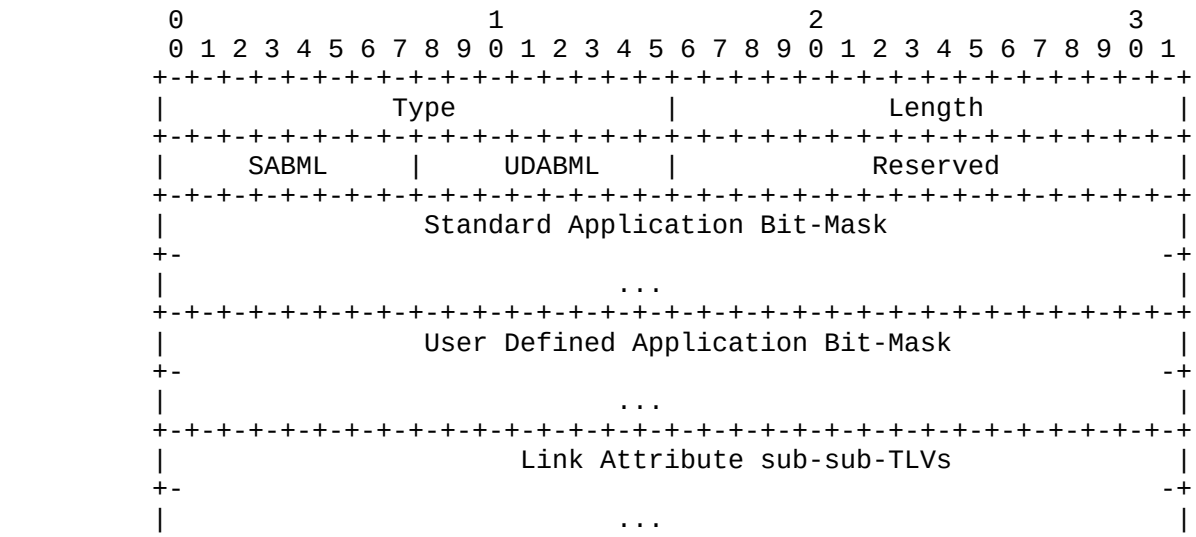
TBD20 - Extended Administrative Group

5. Advertisement of Application Specific Values

Multiple applications can utilize link attributes that are advertised by OSPF. Some examples of applications using the link attributes are Segment Routing Traffic Engineering and LFA [RFC5286].

In some cases the link attribute MAY have different values for different applications. An example could be SRLG [Section 4.1], where values used by LFA could be different then the values used by Segment Routing Traffic Engineering.

To allow advertisement of the application specific values of the link attribute, a new Application Specific Link Attributes (ASLA) sub-TLV is defined. The ASLA sub-TLV is a sub-TLV of the OSPFv2 Extended Link TLV [RFC7471] and OSPFv3 Router-Link TLV [RFC8362]. The ASLA sub-TLV is an optional sub-TLV and can appear multiple times in the OSPFv2 Extended Link TLV and OSPFv3 Router-Link TLV. It has the following format:



where:

Type: TBD21 (OSPFv2), TBD22 (OSPFv3)

Length: variable

SABML: Standard Application Bit-Mask Length. If the Standard Application Bit-Mask is not present, the Standard Application Bit-Mask Length MUST be set to 0.

UDABML: User Defined Application Bit-Mask Length. If the User Defined Application Bit-Mask is not present, the User Defined Application Bit-Mask Length MUST be set to 0.

Standard Application Bit-Mask: Optional set of bits, where each bit represents a single standard application. The following bits are defined by this document:

Bit-0: RSVP Traffic Engineering

Bit-1: Segment Routing Traffic Engineering

Bit-2: Loop Free Alternate (LFA). Includes all LFA types.

Bit-3: Flexible Algorithm as described in [I-D.ietf-lsr-flex-algo].

User Defined Application Bit-Mask: Optional set of bits, where each bit represents a single user defined application.

Standard Application Bits are defined/sent starting with Bit 0. Additional bit definitions that are defined in the future SHOULD be assigned in ascending bit order so as to minimize the number of octets that will need to be transmitted.

User Defined Application bits have no relationship to Standard Application bits and are NOT managed by IANA or any other standards body. It is recommended that bits are used starting with Bit 0 so as to minimize the number of octets required to advertise all of them.

Undefined bits in both Bit-Masks MUST be transmitted as 0 and MUST be ignored on receipt. Bits that are NOT transmitted MUST be treated as if they are set to 0 on receipt.

If the link attribute advertisement is limited to be used by a specific set of applications, corresponding Bit-Masks MUST be present and application specific bit(s) MUST be set for all applications that use the link attributes advertised in the ASLA sub-TLV.

Application Bit-Masks apply to all link attributes that support application specific values and are advertised in the ASLA sub-TLV.

The advantage of not making the Application Bit-Masks part of the attribute advertisement itself is that we can keep the format of the link attributes that have been defined previously and reuse the same format when advertising them in the ASLA sub-TLV.

When neither the Standard Application Bits nor the User Defined Application bits are set (i.e., both SABML and UDABML are 0) in the ASLA sub-TLV, then the link attributes included in it MUST be considered as being applicable to all applications.

If, however, another advertisement of the same link attribute includes any Application Bit-Mask in the ASLA sub-TLV, applications that are listed in the Application Bit-Masks of such ASLA sub-TLV SHOULD use the attribute advertisement which has the application specific bit set in the Application Bit-Masks.

If the same application is listed in the Application Bit-Masks of more then one ASLA sub-TLV, the application SHOULD use the first advertisement and ignore any subsequent advertisements of the same attribute. This situation SHOULD be logged as an error.

This document defines the initial set of link attributes that MUST use ASLA sub-TLV if advertised in the OSPFv2 Extended Link TLV or in the OSPFv3 Router-Link TLV. If the ASLA sub-TLV includes any link attribute(s) NOT listed below, they MUST be ignored. Documents which define new link attributes MUST state whether the new attributes

support application specific values and as such MUST be advertised in an ASLA sub-TLV. The link attributes that MUST be advertised in ASLA sub-TLVs are:

- Shared Risk Link Group
- Unidirectional Link Delay
- Min/Max Unidirectional Link Delay
- Unidirectional Delay Variation
- Unidirectional Link Loss
- Unidirectional Residual Bandwidth
- Unidirectional Available Bandwidth
- Unidirectional Utilized Bandwidth
- Administrative Group
- Extended Administrative Group

6. Maximum Link Bandwidth

Maximum link bandwidth is an application independent attribute of the link that is defined in [RFC3630]. Because it is an application independent attribute, it MUST NOT be advertised in ASLA sub-TLV. Instead, it MAY be advertised as a sub-TLV of the Extended Link Opaque LSA Extended Link TLV in OSPFv2 [RFC7684] or sub-TLV of OSPFv3 E-Router-LSA Router-Link TLV in OSPFv3 [RFC8362].

To advertise the Maximum link bandwidth in the OSPFv2 Extended Link TLV, the same format for sub-TLV defined in [RFC3630] is used with TLV type TBD23.

To advertise the Maximum link bandwidth in the OSPFv3 Router-Link TLV, the same format for sub-TLV defined in [RFC3630] is used with TLV type TBD24.

7. Local Interface IPv6 Address Sub-TLV

The Local Interface IPv6 Address Sub-TLV is an application independent attribute of the link that is defined in [RFC5329]. Because it is an application independent attribute, it MUST NOT be advertised in the ASLA sub-TLV. Instead, it MAY be advertised as a sub-TLV of the OSPFv3 E-Router-LSA Router-Link TLV [RFC8362].

To advertise the Local Interface IPv6 Address Sub-TLV in the OSPFv3 Router-Link TLV, the same format for sub-TLV defined in [RFC5329] is used with TLV type TBD25.

8. Remote Interface IPv6 Address Sub-TLV

The Remote Interface IPv6 Address Sub-TLV is an application independent attribute of the link that is defined in [RFC5329]. Because it is an application independent attribute, it MUST NOT be advertised in the ASLA sub-TLV. Instead, it MAY be advertised as a sub-TLV of the OSPFv3 E-Router-LSA Router-Link TLV [RFC8362].

To advertise the Remote Interface IPv6 Address Sub-TLV in the OSPFv3 Router-Link TLV, the same format for sub-TLV defined in [RFC5329] is used with TLV type TBD26.

9. Deployment Considerations

If link attributes are advertised associated with zero length application bit masks for both standard applications and user defined applications, then that set of link attributes MAY be used by any application. If support for a new application is introduced on any node in a network in the presence of such advertisements, these advertisements MAY be used by the new application. If this is not what is intended, then existing advertisements MUST be readvertised with an explicit set of applications specified before a new application is introduced.

10. Attribute Advertisements and Enablement

This document defines extensions to support the advertisement of application specific link attributes.

Whether the presence of link attribute advertisements for a given application indicates that the application is enabled on that link depends upon the application. Similarly, whether the absence of link attribute advertisements indicates that the application is not enabled depends upon the application.

In the case of RSVP-TE, the advertisement of application specific link attributes implies that RSVP is enabled on that link.

In the case of SRTE, advertisement of application specific link attributes does NOT indicate enablement of SRTE. The advertisements are only used to support constraints which may be applied when specifying an explicit path. SRTE is implicitly enabled on all links which are part of the Segment Routing enabled topology independent of the existence of link attribute advertisements.

In the case of LFA, advertisement of application specific link attributes does NOT indicate enablement of LFA on that link. Enablement is controlled by local configuration.

In the case of Flexible Algorithm, advertisement of application specific link attributes does NOT indicate enablement of Flexible Algorithm on that link. Rather the attributes are used to determine what links are included/excluded in the algorithm specific constrained SPF. This is fully specified in [I-D.ietf-lsr-flex-algo].

If, in the future, additional standard applications are defined to use this mechanism, the specification defining this use MUST define the relationship between application specific link attribute advertisements and enablement for that application.

This document allows the advertisement of application specific link attributes with no application identifiers i.e., both the Standard Application Bit Mask and the User Defined Application Bit Mask are not present Section 5. This supports the use of the link attribute by any application. In the presence of an application where the advertisement of link attribute advertisements is used to infer the enablement of an application on that link (e.g., RSVP-TE), the absence of the application identifier leaves ambiguous whether that application is enabled on such a link. This needs to be considered when making use of the "any application" encoding.

11. Backward Compatibility

Link attributes may be concurrently advertised in both the TE Opaque LSA and the Extended Link Opaque LSA in OSPFv2 and the OSPFv3 Intra-Area-TE-LSA and OSPFv3 Extended LSA Router-Link TLV in OSPFv3.

In fact, there is at least one OSPF implementation that utilizes the link attributes advertised in TE Opaque LSAs [RFC3630] for Non-RSVP TE applications. For example, this implementation of LFA and remote LFA utilizes links attributes such as Shared Risk Link Groups (SRLG) [RFC4203] and Admin Group [[RFC3630] advertised in TE Opaque LSAs. These applications are described in [RFC5286], [RFC7490], [RFC7916] and [RFC8102].

When an OSPF routing domain includes routers using link attributes from the OSPFv2 TE Opaque LSAs or the OSPFv3 Intra-Area-TE-LSA for Non-RSVP TE applications such as LFA, OSPF routers in that domain SHOULD continue to advertise such OSPFv2 TE Opaque LSAs or the OSPFv3 Intra-Area-TE-LSA. If there are also OSPF routers using the link attributes described herein for any other application, OSPF routers in the routing domain will also need to advertise these attributes in

OSPFv2 Extended Link Attributes LSAs or OSPFv3 E-Router-LSA. In such a deployment, the advertised attributes SHOULD be the same and Non-RSVP application access to link attributes is a matter of local policy.

12. Security Considerations

Implementations must assure that malformed TLV and Sub-TLV permutations do not result in errors that cause hard OSPF failures.

13. IANA Considerations

13.1. OSPFv2

OSPFv2 Extended Link TLV Sub-TLVs registry [RFC7684] defines sub-TLVs at any level of nesting for OSPFv2 Extended Link TLVs. This specification updates OSPFv2 Extended Link TLV sub-TLVs registry with the following TLV types:

TBD21 (10 Recommended) - Application Specific Link Attributes

TBD1 (11 Recommended) - Shared Risk Link Group

TBD3 (12 Recommended) - Unidirectional Link Delay

TBD4 (13 Recommended) - Min/Max Unidirectional Link Delay

TBD5 (14 Recommended) - Unidirectional Delay Variation

TBD6 (15 Recommended) - Unidirectional Link Loss

TBD7 (16 Recommended) - Unidirectional Residual Bandwidth

TBD8 (17 Recommended) - Unidirectional Available Bandwidth

TBD9 (18 Recommended) - Unidirectional Utilized Bandwidth

TBD9 (19 Recommended) - Administrative Group

TBD17 (20 Recommended) - Extended Administrative Group

TBD23 (21 Recommended) - Maximum Link Bandwidth

13.2. OSPFv3

OSPFv3 Extended LSA Sub-TLV Registry [RFC8362] defines sub-TLVs at any level of nesting for OSPFv3 Extended LSAs. This specification

updates OSPFv3 Extended LSA Sub-TLV Registry with the following TLV types:

TBD22 (9 Recommended) - Application Specific Link Attributes

TBD2 (10 Recommended) - Shared Risk Link Group

TBD10 (11 Recommended) - Unidirectional Link Delay

TBD11 (12 Recommended) - Min/Max Unidirectional Link Delay

TBD12 (13 Recommended) - Unidirectional Delay Variation

TBD13 (14 Recommended) - Unidirectional Link Loss

TBD14 (15 Recommended) - Unidirectional Residual Bandwidth

TBD15 (16 Recommended) - Unidirectional Available Bandwidth

TBD16 (17 Recommended) - Unidirectional Utilized Bandwidth

TBD19 (18 Recommended) - Administrative Group

TBD20 (19 Recommended) - Extended Administrative Group

TBD24 (20 Recommended) - Maximum Link Bandwidth

TBD25 (21 Recommended) - Local Interface IPv6 Address Sub-TLV

TBD26 (22 Recommended) - Local Interface IPv6 Address Sub-TLV

14. Acknowledgments

Thanks to Chris Bowers for his review and comments.

15. References

15.1. 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>.

[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>.

[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>.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, DOI 10.17487/RFC5714, January 2010, <https://www.rfc-editor.org/info/rfc5714>.

[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>.

[RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", RFC 7684, DOI 10.17487/RFC7684, November 2015, <https://www.rfc-editor.org/info/rfc7684>.

[RFC8362] Lindem, A., Roy, A., Goethals, D., Reddy Vallem, V., and F. Baker, "OSPFv3 Link State Advertisement (LSA) Extensibility", RFC 8362, DOI 10.17487/RFC8362, April 2018, <https://www.rfc-editor.org/info/rfc8362>.

15.2. Informative References

[I-D.ietf-idr-ls-distribution] Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-13 (work in progress), October 2015.

[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-00 (work in progress), May 2018.

[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-25 (work in progress), April 2018.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC4203] Kompella, K., Ed. and Y. Rekhter, Ed., "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4203, DOI 10.17487/RFC4203, October 2005, <https://www.rfc-editor.org/info/rfc4203>.

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[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>.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <https://www.rfc-editor.org/info/rfc7490>.

[RFC7855] Previdi, S., Ed., Filsfils, C., Ed., Decraene, B., Litkowski, S., Horneffer, M., and R. Shakir, "Source Packet Routing in Networking (SPRING) Problem Statement and Requirements", RFC 7855, DOI 10.17487/RFC7855, May 2016, <https://www.rfc-editor.org/info/rfc7855>.

[RFC7916] Litkowski, S., Ed., Decraene, B., Filsfils, C., Raza, K., Horneffer, M., and P. Sarkar, "Operational Management of Loop-Free Alternates", RFC 7916, DOI 10.17487/RFC7916, July 2016, <https://www.rfc-editor.org/info/rfc7916>.

[RFC8102] Sarkar, P., Ed., Hegde, S., Bowers, C., Gredler, H., and S. Litkowski, "Remote-LFA Node Protection and Manageability", RFC 8102, DOI 10.17487/RFC8102, March 2017, <https://www.rfc-editor.org/info/rfc8102>.

[RFC8379] Hegde, S., Sarkar, P., Gredler, H., Nanduri, M., and L. Jalil, "OSPF Graceful Link Shutdown", RFC 8379, DOI 10.17487/RFC8379, May 2018, <https://www.rfc-editor.org/info/rfc8379>.

Authors' Addresses

Peter Psenak (editor)
Cisco Systems, Inc.
Eurovea Centre, Central 3
Pribinova Street 10
Bratislava 81109
Slovakia

Email: ppsenak@cisco.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Les Ginsberg
Cisco Systems
821 Alder Drive
MILPITAS, CA 95035
USA

Email: ginsberg@cisco.com

Wim Henderickx
Nokia
Copernicuslaan 50
Antwerp, 2018 94089
Belgium

Email: wim.henderickx@nokia.com

Jeff Tantsura
Nuage Networks
US

Email: jefftant.ietf@gmail.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Drake
Juniper Networks

Email: jdrake@juniper.net

Internet
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2019

D. Yeung
Arrcus
Y. Qu
Huawei
J. Zhang
Juniper Networks
I. Chen
MITRE Corporation
A. Lindem
Cisco Systems
September 12, 2018

YANG Data Model for OSPF Protocol
draft-ietf-ospf-yang-17

Abstract

This document defines a YANG data model that can be used to configure and manage OSPF. The model is based on YANG 1.1 as defined in RFC 7950 and conforms to the Network Management Datastore Architecture (NDMA) as described in RFC 8342.

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."

This Internet-Draft will expire on March 16, 2019.

Table of Contents

1. Overview

YANG [RFC6020][RFC7950] is a data definition language used to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g., ReST) and encodings other than XML (e.g., JSON) are being defined. Furthermore, YANG data models can be used as the basis for implementation of other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage OSPF and it is an augmentation to the core routing data model. If fully conforms to the Network Management Datastore Architecture (NDMA) [RFC8342]. A core routing data model is defined in [RFC8349], and it provides the basis for the development of data

models for routing protocols. The interface data model is defined in [RFC8343] and is used for referencing interfaces from the routing protocol. The key-chain data model used for OSPF authentication is defined in [RFC8177] and provides both a reference to configured key-chains and an enumeration of cryptographic algorithms.

Both OSPFv2 [RFC2328] and OSPFv3 [RFC5340] are supported. In addition to the core OSPF protocol, features described in other OSPF RFCs are also supported. These includes demand circuit [RFC1793], traffic engineering [RFC3630], multiple address family [RFC5838], graceful restart [RFC3623] [RFC5187], NSSA [RFC3101], and OSPF(v3) as a PE-CE Protocol [RFC4577], [RFC6565]. These non-core features are optional in the OSPF data model.

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. Tree Diagrams

This document uses the graphical representation of data models defined in [RFC8340].

2. Design of Data Model

Although the basis of OSPF configuration elements like routers, areas, and interfaces remains the same, the detailed configuration model varies among router vendors. Differences are observed in terms of how the protocol engine is tied to the routing domain, how multiple protocol engines are be instantiated among others.

The goal of this document is to define a data model that provides a common user interface to the OSPFv2 and OSPFv3 protocols. There is very little information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

2.1. OSPF Operational State

The OSPF operational state is included in the same tree as OSPF configuration consistent with Network Management Datastore Architecture [RFC8342]. Consequently, only the routing container in the ietf-routing model [RFC8349] is augmented. The routing-state container is not augmented.

2.2. Overview

The OSPF YANG module defined in this document has all the common building blocks for the OSPF protocol.

The OSPF YANG module augments the /routing/control-plane-protocols/ control-plane-protocol path defined in the ietf-routing module.

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol:
    +--rw ospf
          .
          .
       +--rw operation-mode?          identityref
       +--rw af?                      identityref
          .
          .
       +--rw areas
       |  +--rw area* [area-id]
       |     +--rw area-id                   area-id-type
       |        .
       |        .
       |     +--rw virtual-links
       |     |  +--rw virtual-link* [transit-area-id router-id]
       |     |     .
       |     |     .
       |     +--rw sham-links {pe-ce-protocol}?
       |     |  +--rw sham-link* [local-id remote-id]
       |     |     .
       |     |     .
       |     +--rw interfaces
       |        +--rw interface* [name]
       |           .
       |           .
       +--rw topologies {multi-topology}?
          +--rw topology* [name]
             .
             .
```

The ospf module is intended to match to the vendor specific OSPF configuration construct that is identified by the local identifier 'name'. The field 'version' allows support for OSPFv2 and OSPFv3.

The ospf container includes one OSPF protocol engine instance. The instance includes OSPF router level configuration and operational state.

The area and area/interface containers respectively define the OSPF configuration and operational state for OSPF areas and interfaces.

The topology container defines the OSPF configuration and operational state for OSPF topologies when the multi-topology feature is supported.

2.3. OSPFv2 and OSPFv3

The data model defined herein supports both OSPFv2 and OSPFv3.

The field 'version' is used to indicate the OSPF version and is mandatory. Based on the configured version, the data model varies to accommodate the differences between OSPFv2 and OSPFv3.

2.4. Optional Features

Optional features are beyond the basic OSPF configuration and it is the responsibility of each vendor to decide whether to support a given feature on a particular device.

This model defines the following optional features:

1. multi-topology: Support Multiple-Topolgy Routing (MTR) [RFC4915].

2. multi-area-adj: Support OSPF multi-area adjacency [RFC5185].

3. explicit-router-id: Support explicit per-instance Router-ID specification.

4. demand-circuit: Support OSPF demand circuits [RFC1793].

5. mtu-ignore: Support disabling OSPF Database Description packet MTU mismatch checking.

6. lls: Support OSPF link-local signaling (LLS) [RFC5613].

7. prefix-suppression: Support OSPF prefix advertisement suppression [RFC6860].

8. ttl-security: Support OSPF Time to Live (TTL) security check suppression [RFC5082].

9. nsr: Support OSPF Non-Stop Routing (NSR).

10. graceful-restart: Support Graceful OSPF Restart [RFC3623], [RFC5187].

11. admin-control: Support Administrative control of the protocol state.

12. auto-cost: Support OSPF interface cost calculation according to reference bandwidth [RFC2328].

13. max-ecmp: Support configuration of the maximum number of Equal-Cost Multi-Path (ECMP) paths.

14. max-lsa: Support configuration of the maximum number of LSAs the OSPF instance will accept [RFC1765].

15. te-rid: Support configuration of the Traffic Engineering (TE) Router-ID [RFC3630], [RFC5329].

16. ldp-igp-sync: Support LDP IGP synchronization [RFC5443].

17. ospfv3-authentication-ipsec: Support IPsec for OSPFv3 authentication [RFC4552].

18. fast-reroute: Support IP Fast Reroute (IP-FRR) [RFC5714].

19. node-flag: Support node-flag for OSPF prefixes. [RFC7684].

20. node-tag: Support node admin tag for OSPF instances [RFC7777].

21. lfa: Support Loop-Free Alternates (LFAs) [RFC5286].

22. remote-lfa: Support Remote Loop-Free Alternates (R-LFA) [RFC7490].

23. stub-router: Support RFC 6987 OSPF Stub Router advertisement [RFC6987].

24. pe-ce-protocol: Support OSPF as a PE-CE protocol [RFC4577], [RFC6565].

25. ietf-spf-delay: Support IETF SPF delay algorithm [RFC8405].

26. bfd: Support BFD detection of OSPF neighbor reachability [RFC5880], [RFC5881], and [I-D.ietf-bfd-yang].

It is expected that vendors will support additional features through vendor-specific augmentations.

2.5. OSPF Router Configuration/Operational State

The ospf container is the top level container in this data model. It represents an OSPF protocol engine instance and contains the router level configuration and operational state. The operational state includes the instance statistics, IETF SPF delay statistics, AS-Scoped Link State Database, local RIB, SPF Log, and the LSA log.

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol:
    +--rw ospf
       .
       .
       +--rw af                     iana-rt-types:address-family
       +--rw explicit-router-id?    rt-types:router-id
       |                            {explicit-router-id}?
       +--rw preference
       |  +--rw (scope)?
       |     +--:(single-value)
       |     |  +--rw all?          uint8
       |     +--:(multi-values)
       |        +--rw (granularity)?
       |        |  +--:(detail)
       |        |  |  +--rw intra-area?   uint8
       |        |  |  +--rw inter-area?   uint8
       |        |  +--:(coarse)
       |        |     +--rw internal?     uint8
       |        +--rw external?     uint8
       +--rw nsr {nsr}?
       |  +--rw enable?   boolean
       +--rw graceful-restart {graceful-restart}?
       |  +--rw enable?                        boolean
       |  +--rw helper-enable?                 boolean
       |  +--rw restart-interval?              uint16
       |  +--rw helper-strict-lsa-checking?    boolean
       +--rw enable?               boolean {admin-control}?
       +--rw auto-cost {auto-cost}?
       |  +--rw enable?                boolean
       |  +--rw reference-bandwidth?   uint32
       +--rw spf-control
       |  +--rw paths?            uint16 {max-ecmp}?
       |  +--rw ietf-spf-delay {ietf-spf-delay}?
       |     +--rw initial-delay?   uint16
       |     +--rw short-delay?     uint16
       |     +--rw long-delay?      uint16
       |     +--rw hold-down?       uint16
       |     +--rw time-to-learn?   uint16
```

```
|     +--ro current-state?             enumeration
|     +--ro remaining-time-to-learn?   uint16
|     +--ro remaining-hold-down?       uint16
|     +--ro last-event-received?       yang:timestamp
|     +--ro next-spf-time?             yang:timestamp
|     +--ro last-spf-time?             yang:timestamp
+--rw database-control
|  +--rw max-lsa?   uint32 {max-lsa}?
+--rw stub-router {stub-router}?
|  +--rw (trigger)?
|     +--:(always)
|        +--rw always!
+--rw mpls
|  +--rw te-rid {te-rid}?
|  |  +--rw ipv4-router-id?   inet:ipv4-address
|  |  +--rw ipv6-router-id?   inet:ipv6-address
|  +--rw ldp
|     +--rw igp-sync?   boolean {ldp-igp-sync}?
+--rw fast-reroute {fast-reroute}?
|  +--rw lfa {lfa}?
+--ro protected-routes
|  +--ro af-stats* [af prefix alternate]
|     +--ro af                    iana-rt-types:address-family
|     +--ro prefix                string
|     +--ro alternate             string
|     +--ro alternate-type?       enumeration
|     +--ro best?                 boolean
|     +--ro non-best-reason?      string
|     +--ro protection-available? bits
|     +--ro alternate-metric1?    uint32
|     +--ro alternate-metric2?    uint32
|     +--ro alternate-metric3?    uint32
+--ro unprotected-routes
|  +--ro af-stats* [af prefix]
|     +--ro af                    iana-rt-types:address-family
|     +--ro prefix                string
+--ro protection-statistics* [frr-protection-method]
|  +--ro frr-protection-method string
|  +--ro af-stats* [af]
|     +--ro af                    iana-rt-types:address-family
|     +--ro total-routes?         uint32
|     +--ro unprotected-routes?   uint32
|     +--ro protected-routes?     uint32
|     +--ro linkprotected-routes? uint32
|     +--ro nodeprotected-routes? uint32
+--rw node-tags {node-tag}?
|  +--rw node-tag* [tag]
|     +--rw tag      uint32
```

```
         +--ro router-id?
         +--ro local-rib
         |  +--ro route* [prefix]
         |     +--ro prefix        inet:ip-prefix
         |     +--ro next-hops
         |     |  +--ro next-hop* [next-hop]
         |     |     +--ro outgoing-interface?   if:interface-ref
         |     |     +--ro next-hop              inet:ip-address
         |     +--ro metric?       uint32
         |     +--ro route-type?   route-type
         |     +--ro route-tag?    uint32
         +--ro statistics
         |  +--ro originate-new-lsa-count?   yang:counter32
         |  +--ro rx-new-lsas-count?         yang:counter32
         |  +--ro as-scope-lsa-count?        yang:gauge32
         |  +--ro as-scope-lsa-chksum-sum?   uint32
         |  +--ro database
         |     +--ro as-scope-lsa-type*
         |        +--ro lsa-type?        uint16
         |        +--ro lsa-count?       yang:gauge32
         |        +--ro lsa-cksum-sum?   int32
         +--ro database
         |  +--ro as-scope-lsa-type* [lsa-type]
         |     +--ro as-scope-lsas
         |        +--ro as-scope-lsa* [lsa-id adv-router]
         |           +--ro lsa-id               union
         |           +--ro adv-router           inet:ipv4-address
         |           +--ro decoded-completed?   boolean
         |           +--ro raw-data?            yang:hex-string
         |           +--ro (version)?
         |              +--:(ospfv2)
         |              |  +--ro ospfv2
         .              .
         .              .
         |              +--:(ospfv3)
         |                 +--ro ospfv3
         .
         .
         +--ro spf-log
         |  +--ro event* [id]
         |     +--ro id                     uint32
         |     +--ro spf-type?              enumeration
         |     +--ro schedule-timestamp?    yang:timestamp
         |     +--ro start-timestamp?       yang:timestamp
               +--ro end-timestamp?         yang:timestamp
         |     +--ro trigger-lsa*
         |        +--ro area-id?      area-id-type
         |        +--ro link-id?      union
```

```
          |        +--ro type?         uint16
          |        +--ro lsa-id?       yang:dotted-quad
          |        +--ro adv-router?   yang:dotted-quad
          |        +--ro seq-num?      uint32
          +--ro lsa-log
          |  +--ro event* [id]
          |     +--ro id                    uint32
          |     +--ro lsa
          |     |  +--ro area-id?      area-id-type
          |     |  +--ro link-id?      union
          |     |  +--ro type?         uint16
          |     |  +--ro lsa-id?       yang:dotted-quad
          |     |  +--ro adv-router?   yang:dotted-quad
          |     |  +--ro seq-num?      uint32
          |     +--ro received-timestamp?   yang:timestamp
          |     +--ro reason?               identityref
          .
          .
```

2.6. OSPF Area Configuration/Operational State

The area container contains OSPF area configuration and the list of interface containers representing all the OSPF interfaces in the area. The area operational state includes the area statistics and the Area Link State Database (LSDB).

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol:
    +--rw ospf
       .
       .
       +--rw areas
       |  +--rw area* [area-id]
       |     +--rw area-id                   area-id-type
       |     +--rw area-type?                identityref
       |     +--rw summary?                  boolean
       |     +--rw default-cost?             uint32
       |     +--rw ranges
       |     |  +--rw range* [prefix]
       |     |     +--rw prefix       inet:ip-prefix
       |     |     +--rw advertise?   boolean
       |     |     +--rw cost?        uint24
       |     +--ro statistics
       |     |  +--ro spf-runs-count?              yang:counter32
       |     |  +--ro abr-count?                   yang:gauge32
       |     |  +--ro asbr-count?                  yang:gauge32
       |     |  +--ro ar-nssa-translator-event-count?
```

```
          |     |                                      yang:counter32
          |     |  +--ro area-scope-lsa-count?        yang:gauge32
          |     |  +--ro area-scope-lsa-cksum-sum?    int32
          |     |  +--ro database
          |     |     +--ro area-scope-lsa-type*
          |     |        +--ro lsa-type?        uint16
          |     |        +--ro lsa-count?       yang:gauge32
          |     |        +--ro lsa-cksum-sum?   int32
          |     +--ro database
          |     |  +--ro area-scope-lsa-type* [lsa-type]
          |     |     +--ro lsa-type           uint16
          |     |     +--ro area-scope-lsas
          |     |        +--ro area-scope-lsa* [lsa-id adv-router]
          |     |           +--ro lsa-id               union
          .     .           .
          .     .           .
          |     |           +--ro (version)?
          |     |              +--:(ospfv2)
          |     |              |  +--ro ospfv2
          |     |              |     +--ro header
          .     .              .     .
          .     .              .     .
          |     |              |     +--ro body
          |     |              |        +--ro router
          .     .              .        .
          .     .              .        .
          |     |              |        +--ro network
          .     .              .        .
          .     .              .        .
          |     |              |        +--ro summary
          .     .              .        .
          .     .              .        .
          |     |              |        +--ro external
          .     .              .        .
          .     .              .        .
          |     |              |        +--ro opaque
          .     .              .        .
          .     .              .        .
          |     |              +--:(ospfv3)
          |     |                 +--ro ospfv3
          |     |                    +--ro header
          .     .                    .
          .     .                    .
          |     |                    +--ro body
          |     |                       +--ro router
          .     .                       .
          .     .                       .
          |     |                       +--ro network
```

```
          .     .                       .
          .     .                       .
          |     |                       +--ro inter-area-prefix
          .     .                       .
          .     .                       .
          |     |                       +--ro inter-area-router
          .     .                       .
          .     .                       .
          |     |                       +--ro as-external
          .     .                       .
          .     .                       .
          |     |                       +--ro nssa
          .     .                       .
          .     .                       .
          |     |                       +--ro link
          .     .                       .
          .     .                       .
          |     |                       +--ro intra-area-prefix
          .     .                       .
          .     .                       .
          |     |                       +--ro router-information
          .     .                       .
          .     .                       .
          |     +--rw virtual-links
          |     |  +--rw virtual-link* [transit-area-id router-id]
          |     |     +--rw transit-area-id       -> ../../../../
          |     |                                    area/area-id
          |     |     +--rw router-id             rt-types:router-id
          |     |     +--rw hello-interval?       uint16
          |     |     +--rw dead-interval?        uint32
          |     |     +--rw retransmit-interval?  uint16
          |     |     +--rw transmit-delay?       uint16
          |     |     +--rw lls?                  boolean {lls}?
          |     |     +--rw ttl-security {ttl-security}?
          |     |     |  +--rw enable?   boolean
          |     |     |  +--rw hops?     uint8
          |     |     +--rw enable?               boolean
          |     |                                 {admin-control}?
          |     |     +--rw authentication
          |     |        +--rw (auth-type-selection)?
          |     |           +--:(auth-ipsec)
          |     |               {ospfv3-authentication-ipsec}?
          |     |           |  +--rw sa?                string
          |     |           +--:(auth-trailer-key-chain)
          |     |           |  +--rw key-chain?
          |     |                     key-chain:key-chain-ref
          |     |           +--:(auth-trailer-key)
          |     |              +--rw key?               string
```

```
          |     |              +--rw crypto-algorithm?   identityref
          |     |     +--ro cost?                 uint16
          |     |     +--ro state?                if-state-type
          |     |     +--ro hello-timer?          uint32
          |     |     +--ro wait-timer?           uint32
          |     |     +--ro dr-router-id?         rt-types:router-id
          |     |     +--ro dr-ip-addr?           inet:ip-address
          |     |     +--ro bdr-router-id?        rt-types:router-id
          |     |     +--ro bdr-ip-addr?          inet:ip-address
          |     |     +--ro statistics
          |     |     |  +--ro if-event-count?        yang:counter32
          |     |     |  +--ro link-scope-lsa-count?  yang:gauge32
          |     |     |  +--ro link-scope-lsa-cksum-sum?
          |     |     |                               uint32
          |     |     |  +--ro database
          |     |     |     +--ro link-scope-lsa-type*
          |     |     |        +--ro lsa-type?        uint16
          |     |     |        +--ro lsa-count?       yang:gauge32
          |     |     |        +--ro lsa-cksum-sum?   int32
          |     |     +--ro neighbors
          |     |     |  +--ro neighbor* [neighbor-router-id]
          |     |     |     +--ro neighbor-router-id
          |     |     |                               rt-types:router-id
          |     |     |     +--ro address?            inet:ip-address
          |     |     |     +--ro dr-router-id?       rt-types:router-id
          |     |     |     +--ro dr-ip-addr?         inet:ip-address
          |     |     |     +--ro bdr-router-id?      rt-types:router-id
          |     |     |     +--ro bdr-ip-addr?        inet:ip-address
          |     |     |     +--ro state?              nbr-state-type
          |     |     |     +--ro dead-timer?         uint32
          |     |     |     +--ro statistics
          |     |     |        +--ro nbr-event-count?
          |     |     |                               yang:counter32
          |     |     |        +--ro nbr-retrans-qlen?
          |     |     |                               yang:gauge32
          |     |     +--ro database
          |     |        +--ro link-scope-lsa-type* [lsa-type]
          |     |           +--ro lsa-type           uint16
          |     |           +--ro link-scope-lsas
          .     .
          .     .
          |     +--rw sham-links {pe-ce-protocol}?
          |     |  +--rw sham-link* [local-id remote-id]
          |     |     +--rw local-id                inet:ip-address
          |     |     +--rw remote-id               inet:ip-address
          |     |     +--rw hello-interval?         uint16
          |     |     +--rw dead-interval?          uint32
          |     |     +--rw retransmit-interval?    uint16
```

```
|     |     +--rw transmit-delay?        uint16
|     |     +--rw lls?                   boolean {lls}?
|     |     +--rw ttl-security {ttl-security}?
|     |     |  +--rw enable?   boolean
|     |     |  +--rw hops?     uint8
|     |     +--rw enable?              boolean
|     |                                {admin-control}?
|     |     +--rw authentication
|     |     |  +--rw (auth-type-selection)?
|     |     |     +--:(auth-ipsec)
|     |     |         {ospfv3-authentication-ipsec}?
|     |     |     |  +--rw sa?                 string
|     |     |     +--:(auth-trailer-key-chain)
|     |     |     |  +--rw key-chain?
|     |     |              key-chain:key-chain-ref
|     |     |     +--:(auth-trailer-key)
|     |     |        +--rw key?                string
|     |     |        +--rw crypto-algorithm?   identityref
|     |     +--rw cost?               uint16
|     |     +--rw mtu-ignore?         boolean
|     |                               {mtu-ignore}?
|     |     +--rw prefix-suppression? boolean
|     |                               {prefix-suppression}?
|     |     +--ro state?              if-state-type
|     |     +--ro hello-timer?        uint32
|     |     +--ro wait-timer?         uint32
|     |     +--ro dr-router-id?       rt-types:router-id
|     |     +--ro dr-ip-addr?         inet:ip-address
|     |     +--ro bdr-router-id?      rt-types:router-id
|     |     +--ro bdr-ip-addr?        inet:ip-address
|     |     +--ro statistics
|     |     |  +--ro if-event-count?        yang:counter32
|     |     |  +--ro link-scope-lsa-count?  yang:gauge32
|     |     |  +--ro link-scope-lsa-cksum-sum?
|     |     |                               uint32
|     |     |  +--ro database
|     |     |     +--ro link-scope-lsa-type*
|     |     |        +--ro lsa-type?        uint16
|     |     |        +--ro lsa-count?       yang:gauge32
|     |     |        +--ro lsa-cksum-sum?   int32
|     |     +--ro neighbors
|     |     |  +--ro neighbor* [neighbor-router-id]
|     |     |     +--ro neighbor-router-id
|     |     |                           rt-types:router-id
|     |     |     +--ro address?        inet:ip-address
|     |     |     +--ro dr-router-id?   rt-types:router-id
|     |     |     +--ro dr-ip-addr?     inet:ip-address
|     |     |     +--ro bdr-router-id?  rt-types:router-id
```

```
          |     |     |     +--ro bdr-ip-addr?    inet:ip-address
          |     |     |     +--ro state?          nbr-state-type
          |     |     |     +--ro dead-timer?     uint32
          |     |     |     +--ro statistics
          |     |     |        +--ro nbr-event-count?
          |     |     |                           yang:counter32
          |     |     |        +--ro nbr-retrans-qlen?
          |     |     |                           yang:gauge32
          |     |     +--ro database
          |     |        +--ro link-scope-lsa-type* [lsa-type]
          |     |           +--ro lsa-type           uint16
          |     |           +--ro link-scope-lsas
          .     .
          .     .
```

2.7. OSPF Interface Configuration/Operational State

The interface container contains OSPF interface configuration and operational state. The interface operational state includes the statistics, list of neighbors, and Link-Local Link State Database (LSDB).

```
module: ietf-ospf
  augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol:
    +--rw ospf
       .
       .
       +--rw areas
       |  +--rw area* [area-id]
       |     .
       |     .
       |     +--rw interfaces
       |        +--rw interface* [name]
       |           +--rw name                    if:interface-ref
       |           +--rw interface-type?         enumeration
       |           +--rw passive?                boolean
       |           +--rw demand-circuit?         boolean
       |                                         {demand-circuit}?
       |           +--rw priority?               uint8
       |           +--rw multi-areas {multi-area-adj}?
       |           |  +--rw multi-area* [multi-area-id]
       |           |     +--rw multi-area-id      area-id-type
       |           |     +--rw cost?              uint16
       |           +--rw static-neighbors
       |           |  +--rw neighbor* [identifier]
       |           |     +--rw identifier       inet:ip-address
       |           |     +--rw cost?            uint16
```

```
|           |     +--rw poll-interval?   uint16
|           |     +--rw priority?        uint8
|           +--rw node-flag?             boolean
|                                        {node-flag}?
|           +--rw bfd {bfd}?
|           |  +--rw enable?   boolean
|           +--rw fast-reroute {fast-reroute}?
|           |  +--rw lfa {lfa}?
|           |     +--rw candidate-enable?   boolean
|           |     +--rw enable?             boolean
|           |     +--rw remote-lfa {remote-lfa}?
|           |        +--rw enable?   boolean
|           +--rw hello-interval?        uint16
|           +--rw dead-interval?         uint32
|           +--rw retransmit-interval?   uint16
|           +--rw transmit-delay?        uint16
|           +--rw lls?                   boolean {lls}?
|           +--rw ttl-security {ttl-security}?
|           |  +--rw enable?   boolean
|           |  +--rw hops?     uint8
|           +--rw enable?                boolean
|                                        {admin-control}?
|           +--rw authentication
|           |  +--rw (auth-type-selection)?
|           |     +--:(auth-ipsec)
|           |     |   {ospfv3-authentication-ipsec}?
|           |     |  +--rw sa?                 string
|           |     +--:(auth-trailer-key-chain)
|           |     |  +--rw key-chain?
|           |     |        key-chain:key-chain-ref
|           |     +--:(auth-trailer-key)
|           |        +--rw key?                string
|           |        +--rw crypto-algorithm?   identityref
|           +--rw cost?               uint16
|           +--rw mtu-ignore?         boolean
|           |                         {mtu-ignore}?
|           +--rw prefix-suppression? boolean
|           |                         {prefix-suppression}?
|           +--ro state?                 if-state-type
|           +--ro hello-timer?        uint32
|           +--ro wait-timer?         uint32
|           +--ro dr-router-id?       rt-types:router-id
|           +--ro dr-ip-addr?         inet:ip-address
|           +--ro bdr-router-id?      rt-types:router-id
|           +--ro bdr-ip-addr?        inet:ip-address
|           +--ro statistics
|           |  +--ro if-event-count?        yang:counter32
|           |  +--ro link-scope-lsa-count?  yang:gauge32
```

```
          |           |  +--ro link-scope-lsa-cksum-sum?
          |           |                                uint32
          |           |  +--ro database
          |           |     +--ro link-scope-lsa-type*
          |           |        +--ro lsa-type?        uint16
          |           |        +--ro lsa-count?       yang:gauge32
          |           |        +--ro lsa-cksum-sum?   int32
          |           +--ro neighbors
          |           |  +--ro neighbor* [neighbor-router-id]
          |           |     +--ro neighbor-router-id
          |           |                               rt-types:router-id
          |           |     +--ro address?            inet:ip-address
          |           |     +--ro dr-router-id?       rt-types:router-id
          |           |     +--ro dr-ip-addr?         inet:ip-address
          |           |     +--ro bdr-router-id?      rt-types:router-id
          |           |     +--ro bdr-ip-addr?        inet:ip-address
          |           |     +--ro state?              nbr-state-type
          |           |     +--ro dead-timer?         uint32
          |           |     +--ro statistics
          |           |        +--ro nbr-event-count?
          |           |                               yang:counter32
          |           |        +--ro nbr-retrans-qlen?
          |           |                               yang:gauge32
          |           +--ro database
          |           .  +--ro link-scope-lsa-type* [lsa-type]
          |           .     +--ro lsa-type              uint16
          |           .     +--ro link-scope-lsas
          .           .
          .           .
          |           +--rw topologies {ospf:multi-topology}?
          |           |  +--rw topology* [name]
          |           |     +--rw name  -> ../../../../../../../../../
          |           |                       ../../../rt:ribs/rib/name
          |           |     +--rw cost? uint32
          |           +--rw instance-id?              uint8
          .
          .
```

2.8. OSPF notification

This YANG model defines a list of notifications that inform YANG clients of important events detected during protocol operation. The defined notifications cover the common set of traps from the OSPFv2 MIB [RFC4750] and OSPFv3 MIB [RFC5643].

```
  notifications:
    +---n if-state-change
    |  +--ro routing-protocol-name?
```

```
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol/name
|  +--ro af?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol
|  +         [rt:name=current()/../routing-protocol-name]/
|  +         ospf:ospf/af
|  +--ro (if-link-type-selection)?
|  |  +--:(interface)
|  |  |  +--ro interface
|  |  |     +--ro interface?   if:interface-ref
|  |  +--:(virtual-link)
|  |  |  +--ro virtual-link
|  |  |     +--ro transit-area-id?      area-id-type
|  |  |     +--ro neighbor-router-id?   rt-types:router-id
|  |  +--:(sham-link)
|  |     +--ro sham-link
|  |        +--ro area-id?          area-id-type
|  |        +--ro local-ip-addr?    inet:ip-address
|  |        +--ro remote-ip-addr?   inet:ip-address
|  +--ro state?                   if-state-type
+---n if-config-error
|  +--ro routing-protocol-name?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol/name
|  +--ro af?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol
|  +         [rt:name=current()/../routing-protocol-name]/
|  +         ospf:ospf/af
|  +--ro (if-link-type-selection)?
|  |  +--:(interface)
|  |  |  +--ro interface
|  |  |     +--ro interface?   if:interface-ref
|  |  +--:(virtual-link)
|  |  |  +--ro virtual-link
|  |  |     +--ro transit-area-id?      area-id-type
|  |  |     +--ro neighbor-router-id?   rt-types:router-id
|  |  +--:(sham-link)
|  |     +--ro sham-link
|  |        +--ro area-id?          area-id-type
|  |        +--ro local-ip-addr?    inet:ip-address
|  |        +--ro remote-ip-addr?   inet:ip-address
|  +--ro packet-source?           yang:dotted-quad
|  +--ro packet-type?             packet-type
|  +--ro error?                   enumeration
+---n nbr-state-change
|  +--ro routing-protocol-name?
```

```
       |  +     -> /rt:routing/control-plane-protocols/
       |  +         control-plane-protocol/name
       |  +--ro af?
       |  +     -> /rt:routing/control-plane-protocols/
       |  +         control-plane-protocol
       |  +         [rt:name=current()/../routing-protocol-name]/
       |  +         ospf:ospf/af
       |  +--ro (if-link-type-selection)?
       |  |  +--:(interface)
       |  |  |  +--ro interface
       |  |  |     +--ro interface?   if:interface-ref
       |  |  +--:(virtual-link)
       |  |  |  +--ro virtual-link
       |  |  |     +--ro transit-area-id?      area-id-type
       |  |  |     +--ro neighbor-router-id?   rt-types:router-id
       |  |  +--:(sham-link)
       |  |     +--ro sham-link
       |  |        +--ro area-id?          area-id-type
       |  |        +--ro local-ip-addr?    inet:ip-address
       |  |        +--ro remote-ip-addr?   inet:ip-address
       |  +--ro neighbor-router-id?      rt-types:router-id
       |  +--ro neighbor-ip-addr?        yang:dotted-quad
       |  +--ro state?                   nbr-state-type
       +---n nbr-restart-helper-status-change
       |  +--ro routing-protocol-name?
       |  +     -> /rt:routing/control-plane-protocols/
       |  +         control-plane-protocol/name
       |  +--ro af?
       |  +     -> /rt:routing/control-plane-protocols/
       |  +         control-plane-protocol
       |  +         [rt:name=current()/../routing-protocol-name]/
       |  +         ospf:ospf/af
       |  +--ro (if-link-type-selection)?
       |  |  +--:(interface)
       |  |  |  +--ro interface
       |  |  |     +--ro interface?   if:interface-ref
       |  |  +--:(virtual-link)
       |  |  |  +--ro virtual-link
       |  |  |     +--ro transit-area-id?      area-id-type
       |  |  |     +--ro neighbor-router-id?   rt-types:router-id
       |  |  +--:(sham-link)
       |  |     +--ro sham-link
       |  |        +--ro area-id?          area-id-type
       |  |        +--ro local-ip-addr?    inet:ip-address
       |  |        +--ro remote-ip-addr?   inet:ip-address
       |  +--ro neighbor-router-id?      rt-types:router-id
       |  +--ro neighbor-ip-addr?        yang:dotted-quad
       |  +--ro status?                  restart-helper-status-type
```

```
|  +--ro age?                      uint32
|  +--ro exit-reason?              restart-exit-reason-type
+---n if-rx-bad-packet
|  +--ro routing-protocol-name?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol/name
|  +--ro af?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol
|  +         [rt:name=current()/../routing-protocol-name]/
|  +         ospf:ospf/af
|  +--ro (if-link-type-selection)?
|  |  +--:(interface)
|  |  |  +--ro interface
|  |  |     +--ro interface?   if:interface-ref
|  |  +--:(virtual-link)
|  |  |  +--ro virtual-link
|  |  |     +--ro transit-area-id?      area-id-type
|  |  |     +--ro neighbor-router-id?   rt-types:router-id
|  |  +--:(sham-link)
|  |     +--ro sham-link
|  |        +--ro area-id?          area-id-type
|  |        +--ro local-ip-addr?    inet:ip-address
|  |        +--ro remote-ip-addr?   inet:ip-address
|  +--ro packet-source?           yang:dotted-quad
|  +--ro packet-type?             packet-type
+---n lsdb-approaching-overflow
|  +--ro routing-protocol-name?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol/name
|  +--ro af?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol
|  +         [rt:name=current()/../routing-protocol-name]/
|  +         ospf:ospf/af
|  +--ro ext-lsdb-limit?          uint32
+---n lsdb-overflow
|  +--ro routing-protocol-name?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol/name
|  +--ro af?
|  +     -> /rt:routing/control-plane-protocols/
|  +         control-plane-protocol
|  +         [rt:name=current()/../routing-protocol-name]/
|  +         ospf:ospf/af
|  +--ro ext-lsdb-limit?          uint32
+---n nssa-translator-status-change
|  +--ro routing-protocol-name?
```

```
      |  +     -> /rt:routing/control-plane-protocols/
      |  +         control-plane-protocol/name
      |  +--ro af?
      |  +     -> /rt:routing/control-plane-protocols/
      |  +         control-plane-protocol
      |  +         [rt:name=current()/../routing-protocol-name]/
      |  +         ospf:ospf/af
      |  +--ro area-id?                 area-id-type
      |  +--ro status?                  nssa-translator-state-type
      +---n restart-status-change
         +--ro routing-protocol-name?
         +     -> /rt:routing/control-plane-protocols/
         +         control-plane-protocol/name
         +--ro af?
         +     -> /rt:routing/control-plane-protocols/
         +         control-plane-protocol
         +         [rt:name=current()/../routing-protocol-name]/
         +         ospf:ospf/af
         +--ro status?                  restart-status-type
         +--ro restart-interval?        uint16
         +--ro exit-reason?             restart-exit-reason-type
```

2.9. OSPF RPC Operations

The "ietf-ospf" module defines two RPC operations:

- clear-database: reset the content of a particular OSPF Link State Database.

- clear-neighbor: restart a particular set of OSPF neighbor.

```
  rpcs:
    +---x clear-neighbor
    |  +---w input
    |     +---w routing-protocol-name
    |     +     -> /rt:routing/control-plane-protocols/
    |     +         control-plane-protocol/name
    |     +---w interface?              if:interface-ref
    +---x clear-database
       +---w input
          +---w routing-protocol-name
                -> /rt:routing/control-plane-protocols/
                   control-plane-protocol/name
```

3. OSPF YANG Module

The following RFCs and drafts are not referenced in the document text but are referenced in the ietf-ospf.yang module: [RFC0905], [RFC4576], [RFC5250], [RFC5881], [RFC6991], [RFC7770], and [RFC8294].

<CODE BEGINS> file "ietf-ospf@2018-08-29.yang"

```
module ietf-ospf {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ospf";

  prefix ospf;

  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-interfaces {
    prefix "if";
    reference "RFC 8343 - A YANG Data Model for Interface
               Management (NDMA Version)";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference "RFC 8294 - Common YANG Data Types for the
               Routing Area";
  }

  import iana-routing-types {
    prefix "iana-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-key-chain {
```

```
      prefix "key-chain";
      reference "RFC 8177 - YANG Data Model for Key Chains";
    }

    import ietf-bfd-types {
      prefix "bfd-types";
      reference "RFC YYYY - YANG Data Model for Bidirectional
        Forwarding Detection (BFD). Please replace YYYY with
        published RFC number for draft-ietf-bfd-yang-17.";
    }

    organization
        "IETF OSPF - OSPF Working Group";

    contact
      "WG Web:   <http://datatracker.ietf.org/group/ospf/>
       WG List:  <mailto:ospf@ietf.org>

       Editor:   Derek Yeung
                 <mailto:derek@arrcus.com>
       Author:   Acee Lindem
                 <mailto:acee@cisco.com>
       Author:   Yingzhen Qu
                 <mailto:yingzhen.qu@huawei.com>
       Author:   Jeffrey Zhang
                 <mailto:zzhang@juniper.net>
       Author:   Ing-Wher Chen
                 <mailto:ing-wher_chen@jabil.com>
       Author:   Dean Bogdanovic
                 <mailto:ivandean@gmail.com>
       Author:   Kiran Agrahara Sreenivasa
                 <mailto:kkoushik@cisco.com>";

    description
       "This YANG module defines the generic configuration and
        operational state for the OSPF protocol common to all
        vendor implementations. It is intended that the module
        will be extended by vendors to define vendor-specific
        OSPF configuration parameters and policies,
        for example route maps or route policies.

        This YANG model conforms to the Network Management
        Datastore Architecture (NDMA) as described in RFC 8242.

        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).

          This version of this YANG module is part of RFC XXXX;
          see the RFC itself for full legal notices.";

      revision 2018-08-29 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: A YANG Data Model for OSPF.";
      }

      feature multi-topology {
        description
          "Support Multiple-Topolgy Routing (MTR).";
        reference "RFC 4915 - Multi-Topology Routing";
      }

      feature multi-area-adj {
        description
          "OSPF multi-area adjacency support as in RFC 5185.";
        reference "RFC 5185 - Multi-Area Adjacency";
      }
      feature explicit-router-id {
        description
          "Set Router-ID per instance explicitly.";
      }

      feature demand-circuit {
        description
          "OSPF demand circuit support as in RFC 1793.";
        reference "RFC 1793 - OSPF Demand Circuits";
      }

      feature mtu-ignore {
        description
          "Disable OSPF Database Description packet MTU
           mismatch checking.";
      }

      feature lls {
        description
          "OSPF link-local signaling (LLS) as in RFC 5613.";
        reference "RFC 5613 - OSPF Link-Local Signaling";
```

```
    }

    feature prefix-suppression {
      description
        "OSPF prefix suppression support as in RFC 6860.";
      reference "RFC 6860 - Hide Transit-Only Networks in OSPF";
    }

    feature ttl-security {
      description
        "OSPF Time to Live (TTL) security check support.";
      reference "RFC 5082 - The Generalized TTL Security
                 Mechanism (GTSM)";
    }

    feature nsr {
      description
        "Non-Stop-Routing (NSR) support.";
    }

    feature graceful-restart {
      description
        "Graceful OSPF Restart as defined in RFC 3623 and
         RFC 5187.";
      reference "RFC 3623 - Graceful OSPF Restart
                 RFC 5187 - OSPFv3 Graceful Restart";
    }

    feature admin-control {
      description
        "Administrative control of the protocol state.";
    }

    feature auto-cost {
      description
        "Calculate OSPF interface cost according to
         reference bandwidth.";
       reference "RFC 2328 - OSPF Version 2";
    }

    feature max-ecmp {
      description
        "Setting maximum number of ECMP paths.";
    }

    feature max-lsa {
      description
        "Setting the maximum number of LSAs the OSPF instance
```

```
          will accept.";
        reference "RFC 1765 - OSPF Database Overload";
      }

      feature te-rid {
        description
          "TE Router-ID.";
        reference "RFC 3630 - Traffic Engineering (TE) Extensions
                   to OSPF Version 2
                   RFC 5329 - Traffic Engineering (TE) Extensions
                   to OSPF Version 3";
      }

      feature ldp-igp-sync {
        description
          "LDP IGP synchronization.";
        reference "RFC 5443 - LDP IGP Synchronization";
      }

      feature ospfv3-authentication-ipsec {
        description
          "Use IPsec for OSPFv3 authentication.";
        reference "RFC 4552 - Authentication/Confidentiality
                  for OSPFv3";
      }

      feature fast-reroute {
       description
         "Support for IP Fast Reroute (IP-FRR).";
       reference "RFC 5714 - IP Fast Rereoute Framework";
      }

      feature node-flag {
       description
         "Support for node-flag for OSPF prefixes.";
       reference "RFC 7684 - OSPFv2 Prefix/Link Advertisement";
      }

      feature node-tag {
        description
          "Support for node admin tag for OSPF routing instances.";
        reference "RFC 7777 - Advertising Node Administrative
                   Tags in OSPF";
      }

      feature lfa {
       description
         "Support for Loop-Free Alternates (LFAs).";
```

```
       reference "RFC 5286 - Basic Specification for IP Fast
                  Reroute: Loop-Free Alternates";
     }

     feature remote-lfa {
      description
        "Support for Remote Loop-Free Alternates (R-LFA).";
      reference "RFC 7490 - Remote Loop-Free Alternate (LFA)
                 Fast Reroute (FRR)";
     }

     feature stub-router {
       description
         "Support for RFC 6987 OSPF Stub Router Advertisement.";
       reference "RFC 6987 - OSPF Stub Router Advertisement";
     }

     feature pe-ce-protocol {
       description
         "Support for OSPF as a PE-CE protocol";
       reference "RFC 4577 - OSPF as the Provider/Customer Edge
                  Protocol for BGP/MPLS IP Virtual Private
                  Networks (VPNs)
                  RFC 6565 - OSPFv3 as a Provider Edge to Customer
                  Edge (PE-CE) Routing Protocol";
     }

     feature ietf-spf-delay {
       description
         "Support for IETF SPF delay algorithm.";
       reference "RFC 8405 - SPF Back-off algorithm for link
                  state IGPs";
     }

     feature bfd {
       description
         "Support for BFD detection of OSPF neighbor reachability.";
       reference "RFC 5880 - Bidirectional Forwarding Detection (BFD)
                  RFC 5881 - Bidirectional Forwarding Detection
                  (BFD) for IPv4 and IPv6 (Single Hop)";
     }

     identity ospf-protocol {
       base "rt:routing-protocol";
       description "Any OSPF protocol version";
     }

     identity ospfv2 {
```

```
      base "ospf-protocol";
      description "OSPFv2 protocol";
    }

    identity ospfv3 {
      base "ospf-protocol";
      description "OSPFv3 protocol";
    }

    identity operation-mode {
      description
        "OSPF operation mode.";
    }

    identity area-type {
      description "Base identity for OSPF area type.";
    }

    identity normal-area {
      base area-type;
      description "OSPF normal area.";
    }

    identity stub-nssa-area {
      base area-type;
      description "OSPF stub or NSSA area.";
    }

    identity stub-area {
      base stub-nssa-area;
      description "OSPF stub area.";
    }

    identity nssa-area {
      base stub-nssa-area;
      description "OSPF Not-So-Stubby Area (NSSA).";
      reference "RFC 3101 - The OSPF Not-So-Stubby Area
                 (NSSA) Option";
    }

    identity ospf-lsa-type {
      description
        "Base identity for OSPFv3 and OSPFv3
         Link State Advertisement (LSA) types";
    }

    identity ospfv2-lsa-type {
      base ospf-lsa-type;
```

```
      description
        "OSPFv2 LSA types";
    }

    identity ospfv2-router-lsa {
      base ospfv2-lsa-type;
      description
        "OSPFv2 Router LSA - Type 1";
    }

    identity ospfv2-network-lsa {
      base ospfv2-lsa-type;
      description
        "OSPFv2 Network LSA - Type 2";
    }

    identity ospfv2-summary-lsa-type {
      base ospfv2-lsa-type;
      description
        "OSPFv2 Summary LSA types";
    }

    identity ospfv2-network-summary-lsa {
      base ospfv2-summary-lsa-type;
      description
        "OSPFv2 Network Summary LSA - Type 3";
    }

    identity ospfv2-asbr-summary-lsa {
      base ospfv2-summary-lsa-type;
      description
        "OSPFv2 AS Boundary Router (ASBR) Summary LSA - Type 4";
    }

    identity ospfv2-external-lsa-type {
      base ospfv2-lsa-type;
      description
        "OSPFv2 External LSA types";
    }

    identity ospfv2-as-external-lsa {
      base ospfv2-external-lsa-type;
      description
        "OSPFv2 AS External LSA - Type 5";
    }

    identity ospfv2-nssa-lsa {
      base ospfv2-external-lsa-type;
```

```
      description
        "OSPFv2 Not-So-Stubby-Area (NSSA) LSA - Type 7";
    }

    identity ospfv2-opaque-lsa-type {
      base ospfv2-lsa-type;
      description
        "OSPFv2 Opaque LSA types";
    }

    identity ospfv2-link-scope-opaque-lsa {
      base ospfv2-opaque-lsa-type;
      description
        "OSPFv2 Link-Scoped Opaque LSA - Type 9";
    }

    identity ospfv2-area-scope-opaque-lsa {
      base ospfv2-opaque-lsa-type;
      description
        "OSPFv2 Area-Scoped Opaque LSA - Type 10";
    }

    identity ospfv2-as-scope-opaque-lsa {
      base ospfv2-opaque-lsa-type;
      description
        "OSPFv2 AS-Scoped Opaque LSA - Type 11";
    }

    identity ospfv2-unknown-lsa-type {
      base ospfv2-lsa-type;
      description
        "OSPFv2 Unknown LSA type";
    }

    identity ospfv3-lsa-type {
      base ospf-lsa-type;
      description
        "OSPFv3 LSA types.";
    }

    identity ospfv3-router-lsa {
      base ospfv3-lsa-type;
      description
        "OSPFv3 Router LSA - Type 0x2001";
    }

    identity ospfv3-network-lsa {
      base ospfv3-lsa-type;
```

```
    description
      "OSPFv3 Network LSA - Type 0x2002";
  }

  identity ospfv3-summary-lsa-type {
    base ospfv3-lsa-type;
    description
      "OSPFv3 Summary LSA types";
  }

  identity ospfv3-inter-area-prefix-lsa {
    base ospfv3-summary-lsa-type;
    description
      "OSPFv3 Inter-area Prefix LSA - Type 0x2003";
  }

  identity ospfv3-inter-area-router-lsa {
    base ospfv3-summary-lsa-type;
    description
      "OSPFv3 Inter-area Router LSA - Type 0x2004";
  }

  identity ospfv3-external-lsa-type {
    base ospfv3-lsa-type;
    description
      "OSPFv3 External LSA types";
  }

  identity ospfv3-as-external-lsa {
    base ospfv3-external-lsa-type;
    description
      "OSPFv3 AS-External LSA - Type 0x4005";
  }

  identity ospfv3-nssa-lsa {
    base ospfv3-external-lsa-type;
    description
      "OSPFv3 Not-So-Stubby-Area (NSSA) LSA - Type 0x2007";
  }

  identity ospfv3-link-lsa {
    base ospfv3-lsa-type;
    description
      "OSPFv3 Link LSA - Type 0x0008";
  }

  identity ospfv3-intra-area-prefix-lsa {
    base ospfv3-lsa-type;
```

```
    description
      "OSPFv3 Intra-area Prefix LSA - Type 0x2009";
  }

  identity ospfv3-router-information-lsa {
    base ospfv3-lsa-type;
    description
      "OSPFv3 Router Information LSA - Types 0x800C,
       0xA00C, and 0xC00C";
  }

  identity ospfv3-unknown-lsa-type {
    base ospfv3-lsa-type;
    description
      "OSPFv3 Unknown LSA type";
  }

  identity lsa-log-reason {
    description
      "Base identity for an LSA log reason.";
  }

  identity lsa-refresh {
    base lsa-log-reason;
    description
      "Identity used when the LSA is logged
       as a result of receiving a refresh LSA.";
  }

  identity lsa-content-change {
    base lsa-log-reason;
    description
      "Identity used when the LSA is logged
       as a result of a change in the content
       of the LSA.";
  }

  identity lsa-purge {
    base lsa-log-reason;
    description
      "Identity used when the LSA is logged
       as a result of being purged.";
  }

  typedef uint24 {
    type uint32 {
      range "0 .. 16777215";
    }
```

```
      description
        "24-bit unsigned integer.";
    }

    typedef area-id-type {
      type yang:dotted-quad;
      description
        "Area ID type.";
    }

    typedef route-type {
      type enumeration {
        enum intra-area {
          description "OSPF intra-area route.";
        }
        enum inter-area {
          description "OSPF inter-area route.";
        }
        enum external-1 {
          description "OSPF type 1 external route.";
        }
        enum external-2 {
          description "OSPF type 2 external route.";
        }
        enum nssa-1 {
          description "OSPF type 1 NSSA route.";
        }
        enum nssa-2 {
          description "OSPF type 2 NSSA route.";
        }
      }
      description "OSPF route type.";
    }

    typedef if-state-type {
      type enumeration {
        enum down {
          value "1";
          description
            "Interface down state.";
        }
        enum loopback {
          value "2";
          description
            "Interface loopback state.";
        }
        enum waiting {
          value "3";
```

```
          description
            "Interface waiting state.";
        }
        enum point-to-point {
          value "4";
          description
            "Interface point-to-point state.";
        }
        enum dr {
          value "5";
          description
            "Interface Designated Router (DR) state.";
        }
        enum bdr {
          value "6";
          description
            "Interface Backup Designated Router (BDR) state.";
        }
        enum dr-other {
          value "7";
          description
            "Interface Other Designated Router state.";
        }
      }
      description
        "OSPF interface state type.";
    }

    typedef nbr-state-type {
      type enumeration {
        enum down {
          value "1";
          description
            "Neighbor down state.";
        }
        enum attempt {
          value "2";
          description
            "Neighbor attempt state.";
        }
        enum init {
          value "3";
          description
            "Neighbor init state.";
        }
        enum 2-way {
          value "4";
          description
```

```
            "Neighbor 2-Way state.";
        }
        enum ex-start {
          value "5";
          description
            "Neighbor exchange start state.";
        }
        enum exchange {
          value "6";
          description
            "Neighbor exchange state.";
        }
        enum loading {
          value "7";
          description
            "Neighbor loading state.";
        }
        enum full {
          value "8";
          description
            "Neighbor full state.";
        }
      }
      description
        "OSPF neighbor state type.";
    }

    typedef restart-helper-status-type {
      type enumeration {
        enum not-helping {
          value "1";
          description
            "Restart helper status not helping.";
        }
        enum helping {
          value "2";
          description
            "Restart helper status helping.";
        }
      }
      description
        "Restart helper status type.";
    }

    typedef restart-exit-reason-type {
      type enumeration {
        enum none {
          value "1";
```

```
          description
            "Restart not attempted.";
        }
        enum in-progress {
          value "2";
          description
            "Restart in progress.";
        }
        enum completed {
          value "3";
          description
            "Restart successfully completed.";
        }
        enum timed-out {
          value "4";
          description
            "Restart timed out.";
        }
        enum topology-changed {
          value "5";
          description
            "Restart aborted due to topology change.";
        }
      }
      description
        "Describes the outcome of the last attempt at a
         graceful restart, either by itself or acting
         as a helper.";
    }

    typedef packet-type {
      type enumeration {
        enum hello {
          value "1";
          description
            "OSPF Hello packet.";
        }
        enum database-descripton {
          value "2";
          description
            "OSPF Database Description packet.";
        }
        enum link-state-request {
          value "3";
          description
            "OSPF Link State Request packet.";
        }
        enum link-state-update {
```

```
          value "4";
          description
            "OSPF Link State Update packet.";
        }
        enum link-state-ack {
          value "5";
          description
            "OSPF Link State Acknowlegement packet.";
        }
      }
      description
        "OSPF packet type.";
    }

    typedef nssa-translator-state-type {
      type enumeration {
        enum enabled {
          value "1";
          description
            "NSSA translator enabled state.";
        }
        enum elected {
          description
            "NSSA translator elected state.";
        }
        enum disabled {
          value "3";
          description
            "NSSA translator disabled state.";
        }
      }
      description
        "OSPF NSSA translator state type.";
    }

    typedef restart-status-type {
      type enumeration {
        enum not-restarting {
          value "1";
          description
            "Router is not restarting.";
        }
        enum planned-restart {
          description
            "Router is going through planned restart.";
        }
        enum unplanned-restart {
          value "3";
```

```
          description
            "Router is going through unplanned restart.";
        }
      }
      description
        "OSPF graceful restart status type.";
    }

    typedef fletcher-checksum16-type {
      type string {
         pattern '(0x)?[0-9a-fA-F]{4}';
      }
       description
        "Fletcher 16-bit checksum in hex-string format 0xXXXX.";
       reference "RFC 905 - ISO Transport Protocol specification
                  ISO DP 8073";
    }

    grouping tlv {
      description
        "Type-Length-Value (TLV)";
      leaf type {
        type uint16;
        description "TLV type.";
      }
      leaf length {
        type uint16;
        description "TLV length (octets).";
      }
      leaf value {
        type yang:hex-string;
        description "TLV value.";
      }
    }

    grouping unknown-tlvs {
      description
        "Unknown TLVs grouping - Used for unknown TLVs or
         unknown sub-TLVs.";
      container unknown-tlvs {
        description "All unknown TLVs.";
        list unknown-tlv {
          description "Unknown TLV.";
          uses tlv;
        }
      }
    }
```

```
    grouping node-tag-tlv {
      description "OSPF Node Admin Tag TLV grouping.";
      list node-tag {
        leaf tag {
          type uint32;
          description
            "Node admin tag value.";
        }
        description
          "List of tags.";
      }
    }

    grouping ospf-router-lsa-flags {
      leaf flags {
        type bits {
          bit V {
            description
              "When set, the router is an endpoint of one or
                 more virtual links.";
          }
          bit E {
            description
              "When set, the router is an AS Boundary Router
                 (ASBR).";
          }
          bit B {
            description
              "When set, the router is an Area Border
                 Router (ABR).";
          }
          bit Nt {
            description
              "When set, the router is an NSSA border router
                 that is unconditionally translating NSSA LSAs
                 into AS-external LSAs.";
          }
        }
        description "Router LSA Flags.";
      }
      description
        "Router LSA Flags - Currently common for OSPFv2 and
         OSPFv3 but it may diverge with future augmentations.";
    }

    grouping ospfv2-router-link {
      description "OSPFv2 router link.";
      leaf link-id {
```

```
        type union {
          type inet:ipv4-address;
          type yang:dotted-quad;
        }
        description "Router-LSA Link ID";
      }
      leaf link-data {
        type union {
          type inet:ipv4-address;
          type uint32;
        }
        description "Router-LSA Link data.";
      }
      leaf type {
        type uint8;
        description "Router-LSA Link type.";
      }
    }

    grouping ospfv2-lsa-body {
      description "OSPFv2 LSA body.";
      container router {
        when "derived-from-or-self(../../header/type, "
            + "'ospf:ospfv2-router-lsa')" {
          description
            "Only applies to Router-LSAs.";
        }
        description
          "Router LSA.";
        uses ospf-router-lsa-flags;
        leaf num-of-links {
          type uint16;
          description "Number of links in Router LSA.";
        }
        container links {
          description "All router Links.";
          list link {
            description "Router LSA link.";
            uses ospfv2-router-link;
            container topologies {
              description "All topologies for the link.";
              list topology {
                description
                  "Topology specific information.";
                leaf mt-id {
                  type uint8;
                  description
                    "The MT-ID for the topology enabled on
```

```
                    the link.";
                }
                leaf metric {
                  type uint16;
                  description "Metric for the topology.";
                }
              }
            }
          }
        }
        container network {
          when "derived-from-or-self(../../header/type, "
              + "'ospfv2-network-lsa')" {
            description
              "Only applies to Network LSAs.";
          }
          description
            "Network LSA.";
          leaf network-mask {
            type inet:ipv4-address;
            description
              "The IP address mask for the network.";
          }
          container attached-routers {
            description "All attached routers.";
            leaf-list attached-router {
              type yang:dotted-quad;
              description
                "List of the routers attached to the network.";
            }
          }
        }
        container summary {
          when "derived-from(../../header/type, "
             + "'ospfv2-summary-lsa-type')" {
            description
              "Only applies to Summary LSAs.";
          }
          description
            "Summary LSA.";
          leaf network-mask {
            type inet:ipv4-address;
            description
              "The IP address mask for the network";
          }
          container topologies {
            description "All topologies for the summary LSA.";
```

```
            list topology {
              description
                "Topology specific information.";
              leaf mt-id {
                type uint8;
                description
                  "The MT-ID for the topology enabled for
                   the summary.";
              }
              leaf metric {
                type uint24;
                description "Metric for the topology.";
              }
            }
          }
        }
        container external {
          when "derived-from(../../header/type, "
             + "'ospfv2-external-lsa-type')" {
            description
              "Only applies to AS-external LSAs and NSSA LSAs.";
          }
          description
            "External LSA.";
          leaf network-mask {
            type inet:ipv4-address;
            description
              "The IP address mask for the network";
          }
          container topologies {
            description "All topologies for the external.";
            list topology {
              description
                "Topology specific information.";
              leaf mt-id {
                type uint8;
                description
                  "The MT-ID for the topology enabled for the
                   external or NSSA prefix.";
              }
              leaf flags {
                type bits {
                  bit E {
                    description
                      "When set, the metric specified is a Type 2
                       external metric.";
                  }
                }
```

```
            description "Flags.";
          }
          leaf metric {
            type uint24;
            description "Metric for the topology.";
          }
          leaf forwarding-address {
            type inet:ipv4-address;
            description
              "Forwarding address.";
          }
          leaf external-route-tag {
            type uint32;
            description
              "Route tag for the topology.";
          }
        }
      }
    }
    container opaque {
      when "derived-from(../../header/type, "
         + "'ospfv2-opaque-lsa-type')" {
        description
          "Only applies to Opaque LSAs.";
      }
      description
        "Opaque LSA.";

      uses unknown-tlvs;

      container node-tag-tlvs {
        description
          "All node tag TLVs.";
        list node-tag-tlv {
          description
            "Node tag TLV.";
          uses node-tag-tlv;
        }
      }

      container router-address-tlv {
        description
          "Router address TLV.";
        leaf router-address {
          type inet:ipv4-address;
          description
            "Router address.";
        }
```

```
        }

        container link-tlvs {
          description "All link TLVs in the LSA.";
          list link-tlv {
            description "Link TLV.";
            leaf link-type {
              type uint8;
              mandatory true;
              description "Link type.";
            }
            leaf link-id {
              type union {
                type inet:ipv4-address;
                type yang:dotted-quad;
              }
              mandatory true;
              description "Link ID.";
            }
            container local-if-ipv4-addrs {
              description "All local interface IPv4 addresses.";
              leaf-list local-if-ipv4-addr {
                type inet:ipv4-address;
                description
                  "List of local interface IPv4 addresses.";
              }
            }
            container remote-if-ipv4-addrs {
              description "All remote interface IPv4 addresses.";
              leaf-list remote-if-ipv4-addr {
                type inet:ipv4-address;
                description
                  "List of remote interface IPv4 addresses.";
              }
            }
            leaf te-metric {
              type uint32;
              description "TE metric.";
            }
            leaf max-bandwidth {
              type rt-types:bandwidth-ieee-float32;
              description "Maximum bandwidth.";
            }
            leaf max-reservable-bandwidth {
              type rt-types:bandwidth-ieee-float32;
              description "Maximum reservable bandwidth.";
            }
            container unreserved-bandwidths {
```

```
            description "All unreserved bandwidths.";
            list unreserved-bandwidth {
              leaf priority {
                type uint8 {
                  range "0 .. 7";
                }
                description "Priority from 0 to 7.";
              }
              leaf unreserved-bandwidth {
                type rt-types:bandwidth-ieee-float32;
                description "Unreserved bandwidth.";
              }
              description
                "List of unreserved bandwidths for different
                 priorities.";
            }
          }
          leaf admin-group {
            type uint32;
            description
              "Administrative group/Resource Class/Color.";
          }
          uses unknown-tlvs;
        }
      }

      container extended-prefix-tlvs {
        description "All extended prefix TLVs in the LSA.";
        list extended-prefix-tlv {
          description "Extended prefix TLV.";
          leaf route-type {
            type enumeration {
              enum unspecified {
                value "0";
                description "Unspecified.";
              }
              enum intra-area {
                value "1";
                description "OSPF intra-area route.";
              }
              enum inter-area {
                value "3";
                description "OSPF inter-area route.";
              }
              enum external {
                value "5";
                description "OSPF External route.";
              }
```

```
                enum nssa {
                  value "7";
                  description "OSPF NSSA external route.";
                }
              }
              description "Route type.";
            }
            leaf flags {
              type bits {
                bit A {
                  description
                    "Attach flag.";
                }
                bit N {
                  description
                    "Node flag.";
                }
              }
              description "Prefix Flags.";
            }
            leaf prefix {
              type inet:ip-prefix;
              description "Address prefix.";
            }
            uses unknown-tlvs;
          }
        }

        container extended-link-tlvs {
          description "All extended link TLVs in the LSA.";
          list extended-link-tlv {
            description "Extended link TLV.";
            uses ospfv2-router-link;
            uses unknown-tlvs;
          }
        }
      }
    }

    grouping ospfv3-lsa-options {
      description "OSPFv3 LSA options";
      leaf options {
        type bits {
          bit AF {
            description
              "When set, the router supprts OSPFv3 AFs as in RFC5838.";
          }
          bit DC {
```

```
            description
              "When set, the router supports demand circuits.";
          }
          bit R {
            description
              "When set, the originator is an active router.";
          }
          bit N {
            description
              "If set, the router is attached to an NSSA";
          }
          bit E {
            description
              "This bit describes the way AS-external LSAs
               are flooded";
          }
          bit V6 {
            description
              "If clear, the router/link should be excluded
               from IPv6 routing calculaton";
          }
        }
        mandatory true;
        description "OSPFv3 LSA options.";
      }
    }

    grouping ospfv3-lsa-prefix {
      description
        "OSPFv3 LSA prefix.";

      leaf prefix {
        type inet:ip-prefix;
        description
          "LSA Prefix.";
      }
      leaf prefix-options {
        type bits {
          bit NU {
            description
              "When set, the prefix should be excluded
               from IPv6 unicast calculations.";
          }
          bit LA {
            description
              "When set, the prefix is actually an IPv6 interface
               address of the Advertising Router.";
          }
```

```
            bit P {
              description
                "When set, the NSSA area prefix should be
                 translated to an AS External LSA and readvertised
                 by the translating NSSA Border Router.";
            }
            bit DN {
              description
                "When set, the inter-area-prefix LSA or
                 AS-external LSA prefix has been advertised as an
                 L3VPN prefix.";
            }
          }
          mandatory true;
          description "Prefix options.";
        }
      }

      grouping ospfv3-lsa-external {
        description
          "AS-External and NSSA LSA.";
        leaf metric {
          type uint24;
          description "Metric";
        }

        leaf flags {
          type bits {
            bit E {
              description
                "When set, the metric specified is a Type 2
                 external metric.";
            }
            bit F {
              description
                "When set, a Forwarding Address is included
                 in the LSA.";
            }
            bit T {
              description
                "When set, an External Route Tag is included
                 in the LSA.";
            }
          }
          description "Flags.";
        }

        leaf referenced-ls-type {
```

```
        type identityref {
          base ospf:ospfv3-lsa-type;
        }
        description "Referenced Link State type.";
      }
      leaf unknown-referenced-ls-type {
        type uint16;
        description
          "Value for an unknown Referenced Link State type.";
      }

      uses ospfv3-lsa-prefix;

      leaf forwarding-address {
        type inet:ipv6-address;
        description
          "Forwarding address.";
      }

      leaf external-route-tag {
        type uint32;
        description
          "Route tag.";
      }
      leaf referenced-link-state-id {
        type yang:dotted-quad;
        description
          "Referenced Link State ID.";
      }
    }

    grouping ospfv3-lsa-body {
      description "OSPFv3 LSA body.";
      container router {
        when "derived-from-or-self(../../header/type, "
           + "'ospfv3-router-lsa')" {
          description
            "Only applies to Router LSAs.";
        }
        description "Router LSA.";
        uses ospf-router-lsa-flags;
        uses ospfv3-lsa-options;

        container links {
          description "All router link.";
          list link {
            description "Router LSA link.";
            leaf interface-id {
```

```
              type uint32;
              description "Interface ID for link.";
            }
            leaf neighbor-interface-id {
              type uint32;
              description "Neighbor's Interface ID for link.";
            }
            leaf neighbor-router-id {
              type rt-types:router-id;
              description "Neighbor's Router ID for link.";
            }
            leaf type {
              type uint8;
              description "Link type: 1 - Point-to-Point Link
                                      2 - Transit Network Link
                                      3 - Stub Network Link Link
                                      4 - Virtual Link";
            }
            leaf metric {
              type uint16;
                description "Link Metric.";
            }
          }
        }
      }
      container network {
        when "derived-from-or-self(../../header/type, "
           + "'ospfv3-network-lsa')" {
          description
            "Only applies to Network LSAs.";
        }
        description "Network LSA.";

        uses ospfv3-lsa-options;

        container attached-routers {
          description "All attached routers.";
          leaf-list attached-router {
            type yang:dotted-quad;
            description
              "List of the routers attached to the network.";
          }
        }
      }
      container inter-area-prefix {
        when "derived-from-or-self(../../header/type, "
           + "'ospfv3-inter-area-prefix-lsa')" {
          description
```

```
            "Only applies to Inter-Area-Prefix LSAs.";
          }
          leaf metric {
            type uint24;
            description "Inter-Area Prefix Metric";
          }
          uses ospfv3-lsa-prefix;
          description "Prefix LSA.";
        }
        container inter-area-router {
          when "derived-from-or-self(../../header/type, "
             + "'ospfv3-inter-area-router-lsa')" {
            description
              "Only applies to Inter-Area-Router LSAs.";
          }
          uses ospfv3-lsa-options;
          leaf metric {
            type uint24;
            description "AS Boundary Router (ASBR) Metric.";
          }
          leaf destination-router-id {
            type rt-types:router-id;
            description
              "The Router ID of the ASBR described by the LSA.";
          }
          description "Inter-Area-Router LSA.";
        }
        container as-external {
          when "derived-from-or-self(../../header/type, "
             + "'ospfv3-as-external-lsa')" {
            description
              "Only applies to AS-external LSAs.";
          }

          uses ospfv3-lsa-external;

          description "AS-External LSA.";
        }
        container nssa {
          when "derived-from-or-self(../../header/type, "
             + "'ospfv3-nssa-lsa')" {
            description
              "Only applies to NSSA LSAs.";
          }
          uses ospfv3-lsa-external;

          description "NSSA LSA.";
        }
```

```
      container link {
        when "derived-from-or-self(../../header/type, "
           + "'ospfv3-link-lsa')" {
          description
            "Only applies to Link LSAs.";
        }
        leaf rtr-priority {
          type uint8;
          description "Router Priority for the interface.";
        }
        uses ospfv3-lsa-options;

        leaf link-local-interface-address {
          type inet:ipv6-address;
          description
            "The originating router's link-local
             interface address for the link.";
        }

        leaf num-of-prefixes {
          type uint32;
          description "Number of prefixes.";
        }

        container prefixes {
          description "All prefixes for the link.";
          list prefix {
            description
              "List of prefixes associated with the link.";
            uses ospfv3-lsa-prefix;
          }
        }
        description "Link LSA.";
      }
      container intra-area-prefix {
        when "derived-from-or-self(../../header/type, "
           + "'ospfv3-intra-area-prefix-lsa')" {
          description
            "Only applies to Intra-Area-Prefix LSAs.";
        }
        description "Intra-Area-Prefix LSA.";

        leaf referenced-ls-type {
          type identityref {
            base ospf:ospfv3-lsa-type;
          }
          description "Referenced Link State type.";
        }
```

```
          leaf unknown-referenced-ls-type {
            type uint16;
            description
              "Value for an unknown Referenced Link State type.";
          }
          leaf referenced-link-state-id {
            type yang:dotted-quad;
            description
              "Referenced Link State ID.";
          }
          leaf referenced-adv-router {
            type rt-types:router-id;
            description
              "Referenced Advertising Router.";
          }

          leaf num-of-prefixes {
            type uint16;
            description "Number of prefixes.";
          }
          container prefixes {
            description "All prefixes in this LSA.";
            list prefix {
              description "List of prefixes in this LSA.";
              uses ospfv3-lsa-prefix;
              leaf metric {
                type uint24;
                description "Prefix Metric.";
              }
            }
          }
        }
        container router-information {
          when "derived-from-or-self(../../header/type, "
             + "'ospfv3-router-information-lsa')" {
            description
              "Only applies to Router Information LSAs (RFC7770).";
          }
          container node-tag-tlvs {
            description
              "All node tag tlvs.";
            list node-tag-tlv {
              description
                "Node tag tlv.";
              uses node-tag-tlv;
            }
          }
          description "Router Information LSA.";
```

```
        reference "RFC 7770 -Extensions for Advertising Router
                   Capabilities";
      }
    }

    grouping lsa-header {
      description
          "Common LSA for OSPFv2 and OSPFv3";
      leaf age {
        type uint16;
        mandatory true;
        description "LSA age.";
      }
      leaf type {
        type identityref {
          base ospf-lsa-type;
        }
        mandatory true;
        description "LSA type";
      }
      leaf adv-router {
        type rt-types:router-id;
        mandatory true;
        description "LSA advertising router.";
      }
      leaf seq-num {
        type uint32;
        mandatory true;
        description "LSA sequence number.";
      }
      leaf checksum {
        type fletcher-checksum16-type;
        mandatory true;
        description "LSA checksum.";
      }
      leaf length {
        type uint16;
        mandatory true;
        description "LSA length including the header.";
      }
    }

    grouping ospfv2-lsa {
      description
          "OSPFv2 LSA - LSAs are uniquely identified by
           the <LSA Type, Link-State ID, Advertising Router>
           tuple with the sequence number differentiating
           LSA instances.";
```

```
      container header {
        must "(derived-from(type, "
           + "'ospfv2-opaque-lsa-type') and "
           + "opaque-id and opaque-type) or "
           + "(not(derived-from(type, "
           + "'ospfv2-opaque-lsa-type')) "
           + "and not(opaque-id) and not(opaque-type))" {
          description
            "Opaque type and ID only apply to Opaque LSAs.";
        }
        description
          "Decoded OSPFv2 LSA header data.";
        leaf option {
          type bits {
            bit MT {
              description
                "When set, the router supports multi-topology as
                 in RFC 4915.";
            }
            bit DC {
              description
                "When set, the router supports demand circuits.";
            }
            bit P {
              description
                "Only used in type-7 LSA. When set, an NSSA
                 border router should translate the type-7 LSA
                 to a type-5 LSA.";
            }
            bit MC {
              description
                "When set, the router supports MOSPF.";
            }
            bit E {
              description
                "This bit describes the way AS-external LSAs
                 are flooded.";
            }
            bit O {
              description
                "When set, the router is opaque-capable as in
                 RFC 5250.";
            }
            bit DN {
              description
                "When a type 3, 5 or 7 LSA is sent from a PE to a CE,
                 the DN bit must be set. See RFC 4576.";
            }
```

```
          }
          mandatory true;
          description "LSA options.";
        }
        leaf lsa-id {
          type yang:dotted-quad;
          mandatory true;
          description "Link-State ID.";
        }

        leaf opaque-type {
          type uint8;
          description "Opaque type.";
        }

        leaf opaque-id {
          type uint24;
          description "Opaque ID.";
        }

        uses lsa-header;
      }
      container body {
        description
          "Decoded OSPFv2 LSA body data.";
        uses ospfv2-lsa-body;
      }
    }

    grouping ospfv3-lsa {
      description
          "Decoded OSPFv3 LSA.";
      container header {
        description
          "Decoded OSPFv3 LSA header data.";
        leaf lsa-id {
          type uint32;
          mandatory true;
          description "OSPFv3 LSA ID.";
        }
        uses lsa-header;
      }
      container body {
        description
          "Decoded OSPF LSA body data.";
        uses ospfv3-lsa-body;
      }
    }
```

```
     grouping lsa-common {
       description
           "Common fields for OSPF LSA represenation.";
       leaf decoded-completed {
         type boolean;
         description
           "The OSPF LSA body is fully decoded.";
       }
       leaf raw-data {
         type yang:hex-string;
         description
           "The complete LSA in network byte
            order hexadecimal as received or originated.";
       }
     }

     grouping lsa {
       description
           "OSPF LSA.";
       uses lsa-common;
       choice version {
         description
           "OSPFv2 or OSPFv3 LSA body.";
         container ospfv2 {
           description "OSPFv2 LSA";
           uses ospfv2-lsa;
         }
         container ospfv3 {
           description "OSPFv3 LSA";
           uses ospfv3-lsa;
         }
       }
     }

     grouping lsa-key {
       description
         "OSPF LSA key.";
       leaf lsa-id {
         type union {
           type yang:dotted-quad;
           type uint32;
         }
         description
           "Link-State ID.";
       }
       leaf adv-router {
         type rt-types:router-id;
         description
```

```
          "Advertising router.";
      }
    }

    grouping instance-stat {
      description "Per-instance statistics";
      leaf originate-new-lsa-count {
        type yang:counter32;
        description "The number of new LSAs originated.";
      }
      leaf rx-new-lsas-count {
        type yang:counter32;
        description "The number of LSAs received.";
      }
      leaf as-scope-lsa-count {
        type yang:gauge32;
        description "The number of AS-scope LSAs.";
      }
      leaf as-scope-lsa-chksum-sum {
        type uint32;
        description
          "The sum of the LSA checksums for AS-scope LSAs.";
      }
      container database {
        description "Container for per AS-scope LSA statistics.";
        list as-scope-lsa-type {
          description "List of AS-scope LSA statistics";
          leaf lsa-type {
            type uint16;
            description "AS-Scope LSA type.";
          }
          leaf lsa-count {
            type yang:gauge32;
            description "The number of LSAs of the LSA type.";
          }
          leaf lsa-cksum-sum {
            type int32;
            description
              "The sum of the LSA checksums of the LSA type.";
          }
        }
      }
    }

    grouping area-stat {
      description "Per-area statistics.";
      leaf spf-runs-count {
        type yang:counter32;
```

```
        description
          "The number of times the intra-area SPF has run.";
      }
      leaf abr-count {
        type yang:gauge32;
        description
          "The total number of Area Border Routers (ABRs)
           reachable within this area.";
      }
      leaf asbr-count {
        type yang:gauge32;
        description
          "The total number of AS Boundary Routers (ASBRs).";
      }
      leaf ar-nssa-translator-event-count {
        type yang:counter32;
        description
          "The number of NSSA translator-state changes.";
      }
      leaf area-scope-lsa-count {
        type yang:gauge32;
        description
          "The number of area-scope LSAs in the area.";
      }
      leaf area-scope-lsa-cksum-sum {
        type int32;
        description "The sum of the area-scope LSAs checksums.";
      }
      container database {
        description "Container for area-scope LSA type statistics.";
        list area-scope-lsa-type {
          description "List of area-scope LSA statistics";
          leaf lsa-type {
            type uint16;
            description "Area-scope LSA type.";
          }
          leaf lsa-count {
            type yang:gauge32;
            description "The number of LSAs of the LSA type.";
          }
          leaf lsa-cksum-sum {
            type int32;
            description
              "The sum of the LSA checksums of the LSA type.";
          }
        }
      }
    }
```

```
    grouping interface-stat {
      description "Per-interface statistics";
      leaf if-event-count {
        type yang:counter32;
        description
          "The number of times this interface has changed its
           state or an error has occurred.";
      }
      leaf link-scope-lsa-count {
        type yang:gauge32;
        description "The number of link-scope LSAs.";
      }
      leaf link-scope-lsa-cksum-sum {
        type uint32;
        description "The sum of link-scope LSA checksums.";
      }
      container database {
        description "Container for link-scope LSA type statistics.";
        list link-scope-lsa-type {
          description "List of link-scope LSA statistics";
          leaf lsa-type {
            type uint16;
            description "Link scope LSA type.";
          }
          leaf lsa-count {
            type yang:gauge32;
            description "The number of LSAs of the LSA type.";
          }
          leaf lsa-cksum-sum {
            type int32;
            description
              "The sum of the LSA checksums of the LSA type.";
          }
        }
      }
    }

    grouping neighbor-stat {
      description "Per-neighbor statistics.";
      leaf nbr-event-count {
        type yang:counter32;
        description
          "The number of times this neighbor has changed
           state or an error has occurred.";
      }
      leaf nbr-retrans-qlen {
        type yang:gauge32;
        description
```

```
          "The current length of the retransmission queue.";
      }
    }

    grouping instance-fast-reroute-config {
      description
        "This group defines global configuration of IP
         Fast ReRoute (FRR).";
      container fast-reroute {
        if-feature fast-reroute;
        description
          "This container may be augmented with global
           parameters for IP-FRR.";
        container lfa {
          if-feature lfa;
          description
            "This container may be augmented with
             global parameters for Loop-Free Alternatives (LFA).
             Container creation has no effect on LFA activation.";
        }
      }
    }

    grouping instance-fast-reroute-state {
      description "IPFRR state data grouping";

      container protected-routes {
        if-feature fast-reroute;
        config false;
        description "Instance protection statistics";

        list af-stats {
          key "af prefix alternate";
          description "Per AF protected prefix information";

          leaf af {
            type iana-rt-types:address-family;
            description
              "Address-family";
          }
          leaf prefix {
            type string;
            description
              "Protected prefix.";
          }
          leaf alternate {
            type string;
            description
```

```
            "Alternate nexthop for the prefix.";
          }
          leaf alternate-type {
            type enumeration {
              enum equal-cost {
                description
                  "ECMP alternate.";
              }
              enum lfa {
                description
                  "LFA alternate.";
              }
              enum remote-lfa {
                description
                  "Remote LFA alternate.";
              }
              enum tunnel {
                description
                  "Tunnel based alternate
                   (like RSVP-TE or GRE).";
              }
              enum ti-lfa {
                description
                  "TI-LFA alternate.";
              }
              enum mrt {
                description
                  "MRT alternate.";
              }
              enum other {
                description
                  "Unknown alternate type.";
              }
            }
            description
              "Type of alternate.";
          }
          leaf best {
            type boolean;
            description
              "Indicates if the alternate is the preferred.";
          }
          leaf non-best-reason {
            type string;
            description
              "Information field to describe why the alternate
               is not best.";
          }
```

```
            leaf protection-available {
              type bits {
                bit node-protect {
                  position 0;
                  description
                    "Node protection available.";
                }
                bit link-protect {
                  position 1;
                  description
                    "Link protection available.";
                }
                bit srlg-protect {
                  position 2;
                  description
                    "SRLG protection available.";
                }
                bit downstream-protect {
                  position 3;
                  description
                    "Downstream protection available.";
                }
                bit other {
                  position 4;
                  description
                    "Other protection available.";
                }
              }
              description "Protection provided by the alternate.";
            }
            leaf alternate-metric1 {
              type uint32;
              description
                "Metric from Point of Local Repair (PLR) to
                 destination through the alternate path.";
            }
            leaf alternate-metric2 {
              type uint32;
              description
                "Metric from PLR to the alternate node";
            }
            leaf alternate-metric3 {
              type uint32;
              description
                "Metric from alternate node to the destination";
            }
          }
        }
```

```
      container unprotected-routes {
        if-feature fast-reroute;
        config false;
        description "List of prefixes that are not protected";

        list af-stats {
          key "af prefix";
          description "Per AF unprotected prefix statistics.";

          leaf af {
            type iana-rt-types:address-family;

            description "Address-family";
          }
          leaf prefix {
            type string;
            description "Unprotected prefix.";
          }
        }
      }

      list protection-statistics {
        key frr-protection-method;
        config false;
        description "List protection method statistics";

        leaf frr-protection-method {
          type string;
          description "Protection method used.";
        }
        list af-stats {
          key af;
          description "Per AF protection statistics.";

          leaf af {
            type iana-rt-types:address-family;
            description "Address-family";
          }
          leaf total-routes {
            type uint32;
            description "Total prefixes.";
          }
          leaf unprotected-routes {
            type uint32;
            description
              "Total prefixes that are not protected.";
          }
          leaf protected-routes {
```

```
            type uint32;
            description
              "Total prefixes that are protected.";
          }
          leaf linkprotected-routes {
            type uint32;
            description
              "Total prefixes that are link protected.";
          }
          leaf nodeprotected-routes {
            type uint32;
            description
              "Total prefixes that are node protected.";
          }
        }
      }
    }

    grouping interface-fast-reroute-config {
      description
        "This group defines interface configuration of IP-FRR.";
      container fast-reroute {
        if-feature fast-reroute;
        container lfa {
          if-feature lfa;
          leaf candidate-enable {
            type boolean;
            default true;
            description
              "Enable the interface to be used as backup.";
          }
          leaf enable {
            type boolean;
            default false;
            description
              "Activates LFA - Per-prefix LFA computation
               is assumed.";
          }
          container remote-lfa {
            if-feature remote-lfa;
            leaf enable {
              type boolean;
              default false;
              description
              "Activates Remote LFA (R-LFA).";
            }
            description
              "Remote LFA configuration.";
```

```
            }
            description
              "LFA configuration.";
          }
          description
            "Interface IP Fast-reroute configuration.";
        }
      }

      grouping interface-physical-link-config {
        description
          "Interface cost configuration that only applies to
           physical interfaces and sham links.";
        leaf cost {
          type uint16 {
            range "1..65535";
          }
          description
            "Interface cost.";
        }
        leaf mtu-ignore {
          if-feature mtu-ignore;
          type boolean;
          description
            "Enable/Disable bypassing the MTU mismatch check in
             Database Description packets.";
        }
        leaf prefix-suppression {
          if-feature prefix-suppression;
          type boolean;
          description
            "Suppress advertisement of the prefixes associated
             with the interface.";
        }
      }

      grouping interface-common-config {
        description
          "Common configuration for all types of interfaces,
           including virtual links and sham links.";

        leaf hello-interval {
          type uint16 {
            range "1..65535";
          }
          units seconds;
          description
            "Interval between hello packets (seconds).";
```

```
      }

      leaf dead-interval {
        type uint32 {
          range "1..2147483647";
        }
        units seconds;
        must "../dead-interval > ../hello-interval" {
          error-message "The dead interval must be "
                      + "larger than the hello interval";
          description
            "The value MUST be greater than 'hello-internval'.";
        }
        description
          "Interval after which a neighbor is declared down
           (seconds) if hello packets are not received.";
      }

      leaf retransmit-interval {
        type uint16 {
          range "1..3600";
        }
        units seconds;
        description
          "Interval between retransmitting unacknowledged Link
           State Advertisements (LSAs) (seconds).";
      }

      leaf transmit-delay {
        type uint16 {
          range "1..3600";
        }
        units seconds;
        description
          "Estimated time needed to transmit Link State Update
           (LSU) packets on the interface (seconds).";
      }

      leaf lls {
        if-feature lls;
        type boolean;
        description
          "Enable/Disable link-local signaling (LLS) support.";
      }

      container ttl-security {
        if-feature ttl-security;
        description "Time to Live (TTL) security check.";
```

```
          leaf enable {
            type boolean;
            description
              "Enable/Disable TTL security check.";
          }
          leaf hops {
            type uint8 {
              range "1..254";
            }
            description
              "Maximum number of hops that an OSPF packet may
               have traversed before reception.";
          }
        }
        leaf enable {
          if-feature admin-control;
          type boolean;
          default true;
          description
            "Enable/disable OSPF protocol on the interface.";
        }

        container authentication {
          description "Authentication configuration.";
          choice auth-type-selection {
            description
              "Options for OSPFv3 authentication configuration.";
            case auth-ipsec {
              when "derived-from-or-self(../../../../../../rt:type, "
                +  "'ospf:ospfv3')" {
                description "Applied to OSPFv3 only.";
              }
              if-feature ospfv3-authentication-ipsec;
              leaf sa {
                  type string;
                  description
                    "Security Association (SA) name.";
              }
            }
            case auth-trailer-key-chain {
              leaf key-chain {
                type key-chain:key-chain-ref;
                 description
                  "key-chain name.";
                }
              }
            case auth-trailer-key {
              leaf key {
```

```
              type string;
              description
                "Key string in ASCII format.";
            }
            leaf crypto-algorithm {
              type identityref {
                base key-chain:crypto-algorithm;
              }
              description
                "Cryptographic algorithm associated with key.";
            }
          }
        }
      }
    }

    grouping interface-config {
      description "Configuration for real interfaces.";

      leaf interface-type {
        type enumeration {
          enum "broadcast" {
            description
              "Specify OSPF broadcast multi-access network.";
          }
          enum "non-broadcast" {
            description
              "Specify OSPF Non-Broadcast Multi-Access
               (NBMA) network.";
          }
          enum "point-to-multipoint" {
            description
              "Specify OSPF point-to-multipoint network.";
          }
          enum "point-to-point" {
            description
              "Specify OSPF point-to-point network.";
          }
        }
        description
          "Interface type.";
      }

      leaf passive {
        type boolean;
        description
          "Enable/Disable passive interface - a passive interface's
           prefix will be advertised but no neighbor adjacencies
```

```
           will be formed on the interface.";
       }

       leaf demand-circuit {
         if-feature demand-circuit;
         type boolean;
         description
           "Enable/Disable demand circuit.";
       }

       leaf priority {
         type uint8;
         description
           "Configure OSPF router priority.";
       }

       container multi-areas {
         if-feature multi-area-adj;
         description "Container for multi-area config.";
         list multi-area {
           key multi-area-id;
           description
             "Configure OSPF multi-area adjacency.";
           leaf multi-area-id {
             type area-id-type;
             description
               "Multi-area adjacency area ID.";
           }
           leaf cost {
             type uint16;
             description
               "Interface cost for multi-area adjacency.";
           }
         }
       }

       container static-neighbors {
         description "Statically configured neighbors.";

         list neighbor {
           key "identifier";
           description
             "Specify a static OSPF neighbor.";

           leaf identifier {
             type inet:ip-address;
             description
               "Neighbor Router ID, IPv4 address, or IPv6 address.";
```

```
          }

          leaf cost {
            type uint16 {
              range "1..65535";
            }
            description "Neighbor cost.";
          }
          leaf poll-interval {
            type uint16 {
              range "1..65535";
            }
            units seconds;
            description
              "Neighbor poll interval (seconds) for sending OSPF
               hello packets to discover the neighbor on NBMA
               networks.";
          }
          leaf priority {
            type uint8 {
              range "1..255";
            }
            description "Neighbor priority for DR election.";
          }
        }
      }

      leaf node-flag {
        if-feature node-flag;
        type boolean;
        default false;
        description
          "Set prefix as identifying the advertising router.";
        reference "RFC 7684 - OSPFv2 Prefix/Link Attribute
                   Advertisement";
      }

      container bfd {
        if-feature bfd;
        description "BFD Client Configuration.";
        uses bfd-types:client-cfg-parms;
        reference "draft-ietf-bfd-yang-xx.txt:
                   YANG Data Model for Bidirectional Forwarding
                   Detection (BFD)";
      }

      uses interface-fast-reroute-config;
      uses interface-common-config;
```

```
      uses interface-physical-link-config;
    }

    grouping neighbor-state {
      description
        "OSPF neighbor operational state.";

      leaf address {
        type inet:ip-address;
        config false;
        description
          "Neighbor address.";
      }
      leaf dr-router-id {
        type rt-types:router-id;
        config false;
        description "Neighbor's Designated Router (DR) Router ID.";
      }

      leaf dr-ip-addr {
        type inet:ip-address;
        config false;
        description "Neighbor's Designated Router (DR) IP address.";
      }

      leaf bdr-router-id {
        type rt-types:router-id;
        config false;
        description
          "Neighbor's Backup Designated Router (BDR) Router ID.";
      }

      leaf bdr-ip-addr {
        type inet:ip-address;
        config false;
        description
          "Neighbor's Backup Designated Router (BDR) IP Address.";
      }
      leaf state {
        type nbr-state-type;
        config false;
        description
          "OSPF neighbor state.";
      }
      leaf dead-timer {
        type uint32;
        units "seconds";
        config false;
```

```
        description "This timer tracks the remaining time before
                     the neighbor is declared dead.";
      }
      container statistics {
        config false;
        description "Per-neighbor statistics";
        uses neighbor-stat;
      }
    }

    grouping interface-common-state {
      description
        "OSPF interface common operational state.";
      reference "RFC2328 Section 9";

      leaf state {
        type if-state-type;
        config false;
        description "Interface state.";
      }

      leaf hello-timer {
        type uint32;
        units "seconds";
        config false;
        description "This timer tracks the remaining time before
                     the next hello packet is sent on the
                     interface.";
      }

      leaf wait-timer {
        type uint32;
        units "seconds";
        config false;
        description "This timer tracks the remaining time before
                     the interface exits the Waiting state.";
      }

      leaf dr-router-id {
        type rt-types:router-id;
        config false;
        description "Designated Router (DR) Router ID.";
      }

      leaf dr-ip-addr {
        type inet:ip-address;
        config false;
        description "Designated Router (DR) IP address.";
```

```
      }

      leaf bdr-router-id {
        type rt-types:router-id;
        config false;
        description "Backup Designated Router (BDR) Router ID.";
      }

      leaf bdr-ip-addr {
        type inet:ip-address;
        config false;
        description "Backup Designated Router (BDR) IP Address.";
      }

      container statistics {
        config false;
        description "Per-interface statistics";
        uses interface-stat;
      }

      container neighbors {
        config false;
        description "All neighbors for the interface.";
        list neighbor {
          key "neighbor-router-id";
          description
            "List of interface OSPF neighbors.";
          leaf neighbor-router-id {
            type rt-types:router-id;
            description
              "Neighbor Router ID.";
          }
          uses neighbor-state;
        }
      }
      container database {
        config false;
        description "Link-scope Link State Database.";
        list link-scope-lsa-type {
          key "lsa-type";
          description
            "List OSPF link-scope LSAs.";
          leaf lsa-type {
            type uint16;
            description "OSPF link-scope LSA type.";
          }
          container link-scope-lsas {
            description
```

```
              "All link-scope LSAs of this LSA type.";
            list link-scope-lsa {
              key "lsa-id adv-router";
              description "List of OSPF link-scope LSAs";
              uses lsa-key;
              uses lsa {
                refine "version/ospfv2/ospfv2" {
                  must "derived-from-or-self( "
                     + "../../../../../../../../../../"
                     + "rt:type, 'ospf:ospfv2')" {
                    description "OSPFv2 LSA.";
                  }
                }
                refine "version/ospfv3/ospfv3" {
                  must "derived-from-or-self( "
                     + "../../../../../../../../../../"
                     + "rt:type, 'ospf:ospfv3')" {
                    description "OSPFv3 LSA.";
                  }
                }
              }
            }
          }
        }
      }
    }

    grouping interface-state {
      description
        "OSPF interface operational state.";
      reference "RFC2328 Section 9";

      uses interface-common-state;
    }

    grouping virtual-link-config {
      description
        "OSPF virtual link configuration state.";

      uses interface-common-config;
    }

    grouping virtual-link-state {
      description
        "OSPF virtual link operational state.";

      leaf cost {
        type uint16 {
```

```
          range "1..65535";
        }
        config false;
        description
          "Virtual link interface cost.";
      }
      uses interface-common-state;
    }

    grouping sham-link-config {
      description
        "OSPF sham link configuration state.";

      uses interface-common-config;
      uses interface-physical-link-config;
    }

    grouping sham-link-state {
      description
        "OSPF sham link operational state.";
      uses interface-common-state;
    }

    grouping af-area-config {
      description
        "OSPF address-family specific area config state.";

      container ranges {
        description "Container for summary ranges";

        list range {
          key "prefix";
          description
            "Summarize routes matching address/mask -
             Applicable to Area Border Routers (ABRs) only.";
          leaf prefix {
            type inet:ip-prefix;
            description
              "IPv4 or IPv6 prefix";
          }
          leaf advertise {
            type boolean;
            description
              "Advertise or hide.";
          }
          leaf cost {
            type uint24 {
              range "0..16777214";
```

```
            }
            description
              "Advertised cost of summary route.";
          }
        }
      }
    }

    grouping area-common-config {
      description
        "OSPF area common configuration state.";

      leaf summary {
        when "derived-from(../area-type,'ospf:stub-nssa-area')" {
          description
            "Summary advertisement into the stub/NSSA area.";
        }
        type boolean;
        description
          "Enable/Disable summary advertisement into the stub or
           NSSA area.";
      }
      leaf default-cost {
        when "derived-from(../area-type,'ospf:stub-nssa-area')" {
            description
              "Cost for LSA default route advertised into the
               stub or NSSA area.";
        }
        type uint32 {
          range "1..16777215";
        }
        description
          "Set the summary default route cost for a
           stub or NSSA area.";
      }
    }

    grouping area-config {
      description
        "OSPF area configuration state.";

      leaf area-type {
        type identityref {
          base area-type;
        }
        default normal-area;
        description
          "Area type.";
```

```
      }

      uses area-common-config;
      uses af-area-config;
    }

    grouping area-state {
      description
        "OSPF area operational state.";

      container statistics {
        config false;
        description "Per-area statistics";
        uses area-stat;
      }

      container database {
        config false;
        description "Area-scope Link State Database.";
        list area-scope-lsa-type {
          key "lsa-type";
          description "List OSPF area-scope LSAs.";
          leaf lsa-type {
            type uint16;
            description "OSPF area-scope LSA type.";
          }
          container area-scope-lsas {
            description
              "All area-scope LSAs of an area-scope
               LSA type.";
            list area-scope-lsa {
              key "lsa-id adv-router";
              description "List of OSPF area-scope LSAs";
              uses lsa-key;
              uses lsa {
                refine "version/ospfv2/ospfv2" {
                  must "derived-from-or-self( "
                     + "../../../../../../../../"
                     + "rt:type, 'ospf:ospfv2')" {
                    description "OSPFv2 LSA.";
                  }
                }
                refine "version/ospfv3/ospfv3" {
                  must "derived-from-or-self( "
                     + "../../../../../../../../"
                     + "rt:type, 'ospf:ospfv3')" {
                    description "OSPFv3 LSA.";
                  }
```

```
                }
              }
            }
          }
        }
      }

      grouping local-rib {
        description "Local-rib - RIB for Routes computed by the local
                     OSPF routing instance.";
        container local-rib {
          config false;
          description "Local-rib.";
          list route {
            key "prefix";
            description "Routes";
            leaf prefix {
              type inet:ip-prefix;
              description "Destination prefix.";
            }
            container next-hops {
              description "Next hops for the route.";
              list next-hop {
                key "next-hop";
                description "List of next hops for the route";
                leaf outgoing-interface {
                  type if:interface-ref;
                  description
                    "Name of the outgoing interface.";
                }
                leaf next-hop {
                 type inet:ip-address;
                 description "Nexthop address.";
                }
              }
            }
            leaf metric {
              type uint32;
              description "Metric for this route.";
            }
            leaf route-type {
              type route-type;
              description "Route type for this route.";
            }
            leaf route-tag {
              type uint32;
              description "Route tag for this route.";
```

```
          }
        }
      }
    }

    grouping ietf-spf-delay {
      leaf initial-delay {
        type uint16;
        units msec;
        description
          "Delay used while in QUIET state (milliseconds).";
      }
      leaf short-delay {
        type uint16;
        units msec;
        description
          "Delay used while in SHORT_WAIT state (milliseconds).";
      }
      leaf long-delay {
        type uint16;
        units msec;
        description
          "Delay used while in LONG_WAIT state (milliseconds).";
      }
      leaf hold-down {
        type uint16;
        units msec;
        description
          "Timer used to consider an IGP stability period
           (milliseconds).";
      }
      leaf time-to-learn {
        type uint16;
        units msec;
        description
          "Duration used to learn all the IGP events
           related to a single component failure (milliseconds).";
      }
      leaf current-state {
        type enumeration {
          enum "quiet" {
            description "QUIET state";
          }
          enum "short-wait" {
            description "SHORT_WAIT state";
          }
          enum "long-wait" {
            description "LONG_WAIT state";
```

```
          }
        }
        config false;
        description
          "Current SPF backoff algorithm state.";
      }
      leaf remaining-time-to-learn {
        type uint16;
        units "seconds";
        config false;
        description
          "Remaining time until time-to-learn timer fires.";
      }
      leaf remaining-hold-down {
        type uint16;
        units "seconds";
        config false;
        description
          "Remaining time until hold-down timer fires.";
      }
      leaf last-event-received {
        type yang:timestamp;
        config false;
        description
          "Time of last SPF triggering event.";
      }
      leaf next-spf-time {
        type yang:timestamp;
        config false;
        description
          "Time when next SPF has been scheduled.";
      }
      leaf last-spf-time {
        type yang:timestamp;
        config false;
        description
          "Time of last SPF computation.";
        }
      description
        "Grouping for IETF SPF delay configuration and state";
    }

    grouping node-tag-config {
      description
        "OSPF node tag config state.";
      container node-tags {
        if-feature node-tag;
        list node-tag {
```

```
          key tag;
          leaf tag {
            type uint32;
              description
                "Node tag value.";
          }
          description
            "List of tags.";
        }
        description
          "Container for node admin tags.";
      }
    }

    grouping instance-config {
      description
        "OSPF instance config state.";

      leaf explicit-router-id {
        if-feature explicit-router-id;
        type rt-types:router-id;
        description
          "Defined in RFC 2328. A 32-bit number
           that uniquely identifies the router.";
      }

      container preference {
        description "Route preference config state.";
        choice scope {
          description
            "Options for expressing preference
             as single or multiple values.";
          case single-value {
            leaf all {
              type uint8;
              description
                "Preference for intra-area, inter-area, and
                 external routes.";
            }
          }
          case multi-values {
            choice granularity {
              description
                "Options for expressing preference
                 for intra-area and inter-area routes.";
              case detail {
                leaf intra-area {
                  type uint8;
```

```
                  description
                    "Preference for intra-area routes.";
                }
                leaf inter-area {
                  type uint8;
                  description
                    "Preference for inter-area routes.";
                }
              }
              case coarse {
                leaf internal {
                  type uint8;
                  description
                    "Preference for both intra-area and
                     inter-area routes.";
                }
              }
            }
            leaf external {
              type uint8;
              description
                "Preference for AS external routes.";
            }
          }
        }
      }

      container nsr {
        if-feature nsr;
        description
          "Non-Stop Routing (NSR) config state.";
        leaf enable {
          type boolean;
          description
            "Enable/Disable NSR.";
        }
      }

      container graceful-restart {
        if-feature graceful-restart;
        description
          "Graceful restart config state.";
        reference "RFC 3623 - OSPF Graceful Restart
                   RFC 5187 - OSPFv3 Graceful Restart";
        leaf enable {
          type boolean;
          description
            "Enable/Disable graceful restart as defined in RFC 3623
```

```
            for OSPFv2 and RFC 5187 for OSPFv3.";
        }
        leaf helper-enable {
          type boolean;
          description
            "Enable graceful restart helper support for restarting
             routers (RFC 3623 Section 3).";
        }
        leaf restart-interval {
          type uint16 {
            range "1..1800";
          }
          units seconds;
          default "120";
          description
            "Interval to attempt graceful restart prior
             to failing (RFC 3623 Section B.1) (seconds)";
        }
        leaf helper-strict-lsa-checking {
          type boolean;
          description
            "Terminate graceful restart when an LSA topology change
             is detected (RFC 3623 Section B.2).";
        }
      }

      leaf enable {
        if-feature admin-control;
        type boolean;
        default true;
        description
          "Enable/Disable the protocol.";
      }

      container auto-cost {
        if-feature auto-cost;
        description
          "Interface Auto-cost configuration state.";
        leaf enable {
          type boolean;
          description
            "Enable/Disable interface auto-cost.";
        }
        leaf reference-bandwidth {
          when "../enable = 'true'" {
            description "Only when auto cost is enabled";
          }
          type uint32 {
```

```
            range "1..4294967";
          }
          units Mbits;
          description
            "Configure reference bandwidth used to automatically
             determine interface cost (Mbits). The cost is the
             reference bandwidth divided by the interface speed
             with 1 being the minimum cost.";
        }
      }

      container spf-control {
        leaf paths {
          if-feature max-ecmp;
          type uint16 {
            range "1..32";
          }
          description
            "Maximum number of Equal-Cost Multi-Path (ECMP) paths.";
        }
        container ietf-spf-delay {
          if-feature ietf-spf-delay;
          uses ietf-spf-delay;
          description
            "IETF SPF delay algorithm configuration.";
        }
        description "SPF calculation control.";
      }

      container database-control {
        leaf max-lsa {
          if-feature max-lsa;
          type uint32 {
            range "1..4294967294";
          }
          description
            "Maximum number of LSAs OSPF the router will accept.";
        }
        description "Database maintenance control.";
      }

      container stub-router {
        if-feature stub-router;
        description "Set maximum metric configuration";

        choice trigger {
          description
            "Specific triggers which will enable stub
```

```
             router state.";
          container always {
            presence
              "Enables unconditional stub router support";
            description
              "Unconditional stub router state (advertise
               transit links with max metric";
          }
        }
      }

      container mpls {
        description
          "OSPF MPLS config state.";
        container te-rid {
          if-feature te-rid;
          description
            "Stable OSPF Router IP Address used for Traffic
             Engineering (TE)";
          leaf ipv4-router-id {
            type inet:ipv4-address;
            description
              "Explicitly configure the TE IPv4 Router ID.";
          }
          leaf ipv6-router-id {
            type inet:ipv6-address;
            description
              "Explicitly configure the TE IPv6 Router ID.";
          }
        }
        container ldp {
          description
            "OSPF MPLS LDP config state.";
          leaf igp-sync {
            if-feature ldp-igp-sync;
            type boolean;
            description
              "Enable LDP IGP synchronization.";
          }
        }
      }
      uses instance-fast-reroute-config;
      uses instance-fast-reroute-state;
      uses node-tag-config;
    }

    grouping instance-state {
      description
```

```
        "OSPF instance operational state.";

      leaf router-id {
        type rt-types:router-id;
        config false;
        description
          "Defined in RFC 2328. A 32-bit number
           that uniquely identifies the router.";
      }

      uses local-rib;

      container statistics {
        config false;
        description "Per-instance statistics";
        uses instance-stat;
      }

      container database {
        config false;
        description "AS-scope Link State Database.";
        list as-scope-lsa-type {
          key "lsa-type";
          description "List OSPF AS-scope LSAs.";
          leaf lsa-type {
            type uint16;
            description "OSPF AS scope LSA type.";
          }
          container as-scope-lsas {
            description "All AS-scope of LSA of this LSA type.";
            list as-scope-lsa {
              key "lsa-id adv-router";
              description "List of OSPF AS-scope LSAs";
              uses lsa-key;
              uses lsa {
                refine "version/ospfv2/ospfv2" {
                  must "derived-from-or-self( "
                     + "../../../../../../"
                     + "rt:type, 'ospf:ospfv2')" {
                    description "OSPFv2 LSA.";
                  }
                }
                refine "version/ospfv3/ospfv3" {
                  must "derived-from-or-self( "
                     + "../../../../../../"
                     + "rt:type, 'ospf:ospfv3')" {
                    description "OSPFv3 LSA.";
                  }
```

```
              }
            }
          }
        }
      }
    }
    uses spf-log;
    uses lsa-log;
  }

  grouping ospf-config {
    description
      "OSPF top configuration state - currently empty.";
  }

  grouping ospf-state {
    description
      "OSPF top operational state - currently empty.";
  }

  grouping multi-topology-area-common-config {
    description
      "OSPF multi-topology area common configuration state.";
    leaf summary {
      when "derived-from("
         + "../../../../../areas/area[area-id=current()/../area-id]/"
         + "area-type, 'stub-nssa-area')" {
        description
          "Summary advertisement into the stub/NSSA area.";
      }
      type boolean;
      description
        "Enable/Disable summary advertisement into the
         topology in the stub or NSSA area.";
    }
    leaf default-cost {
      when "derived-from("
         + "../../../../../areas/area[area-id=current()/../area-id]/"
         + "area-type, 'stub-nssa-area')" {
        description
          "Cost for LSA default route advertised into the
           topology into the stub or NSSA area.";
      }
      type uint32 {
        range "1..16777215";
      }
      description
        "Set the summary default route cost for a
```

```
            stub or NSSA area.";
        }
      }

      grouping multi-topology-area-config {
        description
          "OSPF multi-topology area configuration state.";

        uses multi-topology-area-common-config;
        uses af-area-config;
      }

      grouping multi-topology-area-state {
        description
          "OSPF multi-topology area operational state.";
      }

      grouping multi-topology-config {
        description
          "OSPF multi-topology configuration state.";
      }

      grouping multi-topology-state {
        description
          "OSPF multi-topology operational state.";

        uses local-rib;
      }

      grouping multi-topology-interface-config {
        description
          "OSPF multi-topology configuration state.";

        leaf cost {
           type uint32;
           description
             "Interface cost for this topology.";
        }
      }

      grouping multi-topology-interface-state {
        description
          "OSPF multi-topology operational state.";
      }

      grouping ospfv3-interface-config {
        description
          "OSPFv3 interface specific configuration state.";
```

```
        leaf instance-id {
          type uint8 {
            range "0 .. 31";
          }
          description
            "OSPFv3 instance ID.";
        }
      }

      grouping ospfv3-interface-state {
        description
          "OSPFv3 interface specific operational state.";

        leaf interface-id {
          type uint16;
          config false;
          description
            "OSPFv3 interface ID.";
        }
      }

      grouping lsa-identifiers {
        description
          "The parameters that uniquely identify an LSA.";
        leaf area-id {
          type area-id-type;
          description
            "Area ID";
        }
        leaf type {
          type uint16;
          description
            "LSA type.";
        }
        leaf lsa-id {
          type yang:dotted-quad;
          description "Link-State ID.";
        }
        leaf adv-router {
          type rt-types:router-id;
          description
            "LSA advertising router.";
        }
        leaf seq-num {
          type uint32;
          description
            "LSA sequence number.";
        }
```

```
    }

    grouping spf-log {
      description
        "Grouping for SPF log.";
      container spf-log {
        config false;
        description
          "This container lists the SPF log.";
        list event {
          key id;
          description
            "List of SPF log entries represented
             as a wrapping buffer.";
          leaf id {
            type uint32;
            description
              "Event identifier - Ppurely internal value.";
          }
          leaf spf-type {
            type enumeration {
              enum full {
                description
                  "SPF computation was a Full SPF.";
              }
              enum intra {
                description
                  "SPF computation was only for intra-area routes.";
              }
              enum inter {
                description
                  "SPF computation was only for inter-area
                   summary routes.";
              }
              enum external {
                description
                  "SPF computation was only for AS external routes.";
              }
            }
            description
              "The SPF computation type for the SPF log entry.";
          }
          leaf schedule-timestamp {
            type yang:timestamp;
            description
              "This is the timestamp when the computation was
               scheduled.";
          }
```

```
          leaf start-timestamp {
            type yang:timestamp;
            description
              "This is the timestamp when the computation was
               started.";
          }
          leaf end-timestamp {
            type yang:timestamp;
            description
              "This the timestamp when the computation was
               completed.";
          }
          list trigger-lsa {
            description
              "The list of LSAs that triggered the computation.";
            uses lsa-identifiers;
          }
        }
      }
    }

    grouping lsa-log {
      description
        "Grouping for the LSA log.";
      container lsa-log {
        config false;
        description
          "This conatiner lists the LSA log.
           Local LSA modifications are also included
           in the list.";
        list event {
          key id;
          description
            "List of LSA log entries represented
             as a wrapping buffer.";
          leaf id {
            type uint32;
            description
              "Event identifier - purely internal value.";
          }
          container lsa {
            description
              "This container describes the logged LSA.";
            uses lsa-identifiers;
          }
          leaf received-timestamp {
            type yang:timestamp;
            description
```

```
              "This is the timestamp when the LSA was received.
               In case of local LSA update, the timestamp refers
               to the LSA origination time.";
          }
          leaf reason {
            type identityref {
              base lsa-log-reason;
            }
            description
              "This reason for the LSA log entry.";
          }
        }
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol" {
       when "derived-from(rt:type, 'ospf:ospf-protocol')" {
        description
          "This augmentation is only valid for a routing protocol
           instance of OSPF (type 'ospfv2' or 'ospfv3').";
      }
      description "OSPF protocol ietf-routing module
                   control-plane-protocol augmentation.";

      container ospf {
        description
          "OSPF protocol Instance";

        uses ospf-config;
        uses ospf-state;

        leaf af {
          type iana-rt-types:address-family;
          description
            "Address-family of the instance.";
        }

        uses instance-config;
        uses instance-state;

        container areas {
          description "All areas.";
          list area {
            key "area-id";
            description
              "List of OSPF areas";
            leaf area-id {
```

```
            type area-id-type;
            description
              "Area ID";
          }

          uses area-config;
          uses area-state;

          container virtual-links {
            when "derived-from-or-self(../area-type, 'normal-area') "
               + "and ../area-id = '0.0.0.0'" {
              description
                "Virtual links must be in backbone area.";
              }
            description "All virtual links.";
            list virtual-link {
              key "transit-area-id router-id";
              description
                "OSPF virtual link";
              leaf transit-area-id {
                type leafref {
                  path "../../../../area/area-id";
                }
                must "derived-from-or-self("
                  + "../../../../area[area-id=current()]/area-type, "
                  + "'normal-area') and "
                  + "../../../../area[area-id=current()]/area-id != "
                  + "'0.0.0.0'" {
                  error-message "Virtual link transit area must "
                              + "be non-zero.";
                  description
                    "Virtual-link trasit area must be
                     non-zero area.";
                }
                description
                  "Virtual link tranist area ID.";
              }
              leaf router-id {
                type rt-types:router-id;
                description
                  "Virtual Link remote endpoint Router ID.";
              }

              uses virtual-link-config;
              uses virtual-link-state;
            }
          }
          container sham-links {
```

```
              if-feature pe-ce-protocol;
              description "All sham links.";
              list sham-link {
                key "local-id remote-id";
                description
                  "OSPF sham link";
                leaf local-id {
                  type inet:ip-address;
                  description
                    "Address of the local sham Link endpoint.";
                }
                leaf remote-id {
                  type inet:ip-address;
                  description
                    "Address of the remote sham Link endpoint.";
                }
                uses sham-link-config;
                uses sham-link-state;
              }
            }
            container interfaces {
              description "All interfaces.";
              list interface {
                key "name";
                description
                  "List of OSPF interfaces.";
                leaf name {
                  type if:interface-ref;
                  description
                    "Interface name reference.";
                }
                uses interface-config;
                uses interface-state;
              }
            }
          }
        }
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/ospf:ospf" {
      when "derived-from(../rt:type, 'ospf:ospf-protocol')" {
        description
          "This augmentation is only valid for OSPF
           (type 'ospfv2' or 'ospfv3').";
      }
      if-feature multi-topology;
```

```
      description
        "OSPF multi-topology instance configuration
         state augmentation.";
      container topologies {
        description "All topologies.";
        list topology {
          key "name";
          description
            "OSPF topology - The OSPF topology address-family
             must coincide with the routing-instance
             address-family.";
          leaf name {
            type leafref {
              path "../../../../../../rt:ribs/rt:rib/rt:name";
            }
            description "RIB name corresponding to the OSPF
                         topology.";
          }

          uses multi-topology-config;
          uses multi-topology-state;

          container areas {
            description "All areas in the topology.";
            list area {
              key "area-id";
              description
                "List of OSPF areas";
              leaf area-id {
                type area-id-type;
                description
                  "Area ID.";
              }
              uses multi-topology-area-config;
              uses multi-topology-area-state;
            }
          }
        }
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/ospf:ospf/"
          + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface" {
      when "derived-from-or-self(../../../../../rt:type, "
         + "'ospf:ospfv2')" {
        description
          "This augmentation is only valid for OSPFv2.";
```

```
      }
      if-feature ospf:multi-topology;
      description
        "OSPF multi-topology interface configuration state
         augmentation.";
      container topologies {
        description "All topologies for the interface.";
        list topology {
          key "name";
          description "OSPF interface topology.";
          leaf name {
            type leafref {
              path "../../../../../../../../../../"
                 + "rt:ribs/rt:rib/rt:name";
            }
            description
              "Single topology enabled on this interface.";
          }

          uses multi-topology-interface-config;
          uses multi-topology-interface-state;
        }
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/ospf:ospf/"
          + "ospf:areas/ospf:area/ospf:interfaces/ospf:interface" {
      when "derived-from-or-self(../../../../../rt:type, "
         + "'ospf:ospfv3')" {
        description
          "This augmentation is only valid for OSPFv3.";
      }
      description
        "OSPFv3 interface specific configuration state
         augmentation.";
      uses ospfv3-interface-config;
      uses ospfv3-interface-state;
    }

    grouping route-content {
      description
        "This grouping defines OSPF-specific route attributes.";
      leaf metric {
        type uint32;
        description "OSPF route metric.";
      }
      leaf tag {
```

```
        type uint32;
        default "0";
        description "OSPF route tag.";
      }
      leaf route-type {
        type route-type;
        description "OSPF route type";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route" {
      when "derived-from(rt:source-protocol, 'ospf:ospf-protocol')" {
        description
          "This augmentation is only valid for a routes whose
           source protocol is OSPF.";
      }
      description
        "OSPF-specific route attributes.";
      uses route-content;
    }

    /*
     * RPCs
     */

    rpc clear-neighbor {
      description
        "This RPC request clears a particular set of OSPF neighbors.
         If the operation fails for OSPF internal reason, then
         error-tag and error-app-tag should be set to a meaningful
         value.";
      input {
        leaf routing-protocol-name {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
               + "rt:control-plane-protocol/rt:name";
          }
          mandatory "true";
          description
            "OSPF protocol instance which information for neighbors
             are to be cleared.

             If the referenced OSPF instance doesn't exist, then
             this operation SHALL fail with error-tag 'data-missing'
             and error-app-tag
             'routing-protocol-instance-not-found'.";
        }
```

```
        leaf interface {
          type if:interface-ref;
            description
              "Name of the OSPF interface for which neighbors are to
               be cleared.

               If the referenced OSPF interface doesn't exist, then
               this operation SHALL fail with error-tag
               'data-missing'  and error-app-tag
               'ospf-interface-not-found'.";
        }
      }
    }

    rpc clear-database {
      description
        "This RPC request clears a particular OSPF Link State
         Database. If the operation fails for OSPF internal reason,
         then error-tag and error-app-tag should be set to a
         meaningful value.";
      input {
        leaf routing-protocol-name {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
               + "rt:control-plane-protocol/rt:name";
          }
          mandatory "true";
          description
            "OSPF protocol instance whose Link State Database is to
             be cleared.

             If the referenced OSPF instance doesn't exist, then
             this operation SHALL fail with error-tag 'data-missing'
             and error-app-tag
             'routing-protocol-instance-not-found'.";
        }
      }
    }

    /*
     * Notifications
     */

    grouping notification-instance-hdr {
      description
       "This grouping describes common instance specific
        data for OSPF notifications.";
```

```
      leaf routing-protocol-name {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        must "derived-from( "
          + "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol[rt:name=current()]/"
          + "rt:type, 'ospf:ospf-protocol')";
        description
         "OSPF routing protocol instance name.";
      }

      leaf af {
        type leafref {
          path "/rt:routing/"
             + "rt:control-plane-protocols/rt:control-plane-protocol"
             + "[rt:name=current()/../routing-protocol-name]/"
             + "ospf:ospf/af";
        }
        description
          "Address family of the OSPF instance.";
      }
    }

    grouping notification-interface {
      description
        "This grouping provides interface information
         for the OSPF interface specific notification.";

      choice if-link-type-selection {
        description
          "Options for link type.";
        container interface {
          description "Normal interface.";
          leaf interface {
            type if:interface-ref;
            description "Interface.";
          }
        }
        container virtual-link {
          description "virtual-link.";
          leaf transit-area-id {
            type area-id-type;
            description "Area ID.";
          }
          leaf neighbor-router-id {
            type rt-types:router-id;
```

```
            description "Neighbor Router ID.";
          }
        }
        container sham-link {
          description "sham link.";
          leaf area-id {
            type area-id-type;
            description "Area ID.";
          }
          leaf local-ip-addr {
            type inet:ip-address;
            description "Sham link local address.";
          }
          leaf remote-ip-addr {
            type inet:ip-address;
            description "Sham link remote address.";
          }
        }
      }
    }

    grouping notification-neighbor {
      description
        "This grouping provides the neighbor information
         for neighbor specific notifications.";

      leaf neighbor-router-id {
        type rt-types:router-id;
        description "Neighbor Router ID.";
      }

      leaf neighbor-ip-addr {
        type yang:dotted-quad;
        description "Neighbor address.";
      }
    }

    notification if-state-change {
      uses notification-instance-hdr;
      uses notification-interface;

      leaf state {
        type if-state-type;
        description "Interface state.";
      }
      description
        "This notification is sent when an interface
         state change is detected.";
```

```
    }

    notification if-config-error {
      uses notification-instance-hdr;
      uses notification-interface;

      leaf packet-source {
        type yang:dotted-quad;
        description "Source address.";
      }

      leaf packet-type {
        type packet-type;
        description "OSPF packet type.";
      }

      leaf error {
        type enumeration {
          enum "bad-version" {
            description "Bad version.";
          }
          enum "area-mismatch" {
            description "Area mistmatch.";
          }
          enum "unknown-nbma-nbr" {
            description "Unknown NBMA neighbor.";
          }
          enum "unknown-virtual-nbr" {
            description "Unknown virtual link neighbor.";
          }
          enum "auth-type-mismatch" {
            description "Auth type mismatch.";
          }
          enum "auth-failure" {
            description "Auth failure.";
          }
          enum "net-mask-mismatch" {
            description "Network mask mismatch.";
          }
          enum "hello-interval-mismatch" {
            description "Hello interval mismatch.";
          }
          enum "dead-interval-mismatch" {
            description "Dead interval mismatch.";
          }
          enum "option-mismatch" {
            description "Option mismatch.";
          }
```

```
            enum "mtu-mismatch" {
              description "MTU mismatch.";
            }
            enum "duplicate-router-id" {
              description "Duplicate Router ID.";
            }
            enum "no-error" {
              description "No error.";
            }
          }
          description "Error code.";
        }
        description
          "This notification is sent when an interface
           config error is detected.";
      }

      notification nbr-state-change {
        uses notification-instance-hdr;
        uses notification-interface;
        uses notification-neighbor;

        leaf state {
          type nbr-state-type;
          description "Neighbor state.";
        }

        description
          "This notification is sent when aa neighbor
           state change is detected.";
      }

      notification nbr-restart-helper-status-change {
        uses notification-instance-hdr;
        uses notification-interface;
        uses notification-neighbor;

        leaf status {
          type restart-helper-status-type;
          description "Restart helper status.";
        }

        leaf age {
          type uint32;
          units seconds;
          description
            "Remaining time in current OSPF graceful restart
             interval when the router is acting as a restart
```

```
            helper for the neighbor.";
        }

        leaf exit-reason {
          type restart-exit-reason-type;
          description
            "Restart helper exit reason.";
        }
        description
          "This notification is sent when a neighbor restart
           helper status change is detected.";
      }

      notification if-rx-bad-packet {
        uses notification-instance-hdr;
        uses notification-interface;

        leaf packet-source {
          type yang:dotted-quad;
          description "Source address.";
        }

        leaf packet-type {
          type packet-type;
          description "OSPF packet type.";
        }

        description
          "This notification is sent when an OSPF packet that
           cannot be parsed is received on an OSPF interface.";
      }

      notification lsdb-approaching-overflow {
        uses notification-instance-hdr;

        leaf ext-lsdb-limit {
          type uint32;
          description
            "The maximum number of non-default AS-external LSAs
             entries that can be stored in the Link State Database.";
        }

        description
          "This notification is sent when the number of LSAs
           in the router's Link State Database has exceeded
           ninety percent of the AS-external limit (ext-lsdb-limit).";
      }
```

```
    notification lsdb-overflow {
      uses notification-instance-hdr;

      leaf ext-lsdb-limit {
        type uint32;
        description
          "The maximum number of non-default AS-external LSAs
           entries that can be stored in the Link State Database.";
      }

      description
        "This notification is sent when the number of LSAs
         in the router's Link State Database has exceeded the
         AS-external limit (ext-lsdb-limit).";
    }

    notification nssa-translator-status-change {
      uses notification-instance-hdr;

      leaf area-id {
        type area-id-type;
        description "Area ID.";
      }

      leaf status {
        type nssa-translator-state-type;
        description
          "NSSA translator status.";
      }

      description
        "This notification is sent when there is a change
         in the router's role in translating OSPF NSSA LSAs
         to OSPF AS-External LSAs.";
    }

    notification restart-status-change {
      uses notification-instance-hdr;

      leaf status {
        type restart-status-type;
        description
          "Restart status.";
      }

      leaf restart-interval {
        type uint16 {
          range "1..1800";
```

```
        }
        units seconds;
        default "120";
        description
          "Restart interval.";
      }

      leaf exit-reason {
        type restart-exit-reason-type;
        description
          "Restart exit reason.";
      }

      description
        "This notification is sent when the graceful restart
         state for the router has changed.";
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The YANG modules specified in this document define 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 NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in ietf-ospf.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. For OSPF, the ability to modify OSPF configuration will allow the entire OSPF domain to be compromised including peering with unauthorized routers to misroute traffic or mount a massive Denial-of-Service (DoS) attack. The security considerations of OSPFv2 [RFC2328] and [RFC5340] apply to the ietf-ospf.yang module as well.

Some of the readable data nodes in the ietf-ospf.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. The exposure of the Link State Database (LSDB) will expose the detailed topology of the network. This may be undesirable since both due to the fact that exposure may facilitate other attacks. Additionally, network operators may consider their topologies to be sensitive confidential data.

For OSPF authentication, configuration is supported via the specification of key-chains [RFC8177] or the direct specification of key and authentication algorithm. Hence, authentification configuration using the "auth-table-trailer" case in the "authentication" container inherits the security considerations of [RFC8177]. This includes the considerations with respect to the local storage and handling of authentication keys.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. The OSPF YANG module support the "clear-neighbor" and "clear-database" RPCs. If access too either of these is compromised, they can result in temporary network outages be employed to mount DoS attacks.

5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
URI: urn:ietf:params:xml:ns:yang:ietf-ospf
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name: ietf-ospf
namespace: urn:ietf:params:xml:ns:yang:ietf-ospf
prefix: ospf
reference: RFC XXXX
```

6. Acknowledgements

The authors wish to thank Yi Yang, Alexander Clemm, Gaurav Gupta, Ladislav Lhotka, Stephane Litkowski, Greg Hankins, Manish Gupta and Alan Davey for their thorough reviews and helpful comments.

Thanks to Tom Petch for last call review and improvement of the document organization.

This document was produced using Marshall Rose’s xml2rfc tool.

7. References

7.1. Normative References

[I-D.ietf-bfd-yang]
Rahman, R., Zheng, L., Jethanandani, M., Networks, J., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-17 (work in progress), August 2018.

[RFC1793] Moy, J., "Extending OSPF to Support Demand Circuits", RFC 1793, DOI 10.17487/RFC1793, April 1995, <https://www.rfc-editor.org/info/rfc1793>.

[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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", RFC 3101, DOI 10.17487/RFC3101, January 2003, <https://www.rfc-editor.org/info/rfc3101>.

[RFC3623] Moy, J., Pillay-Esnault, P., and A. Lindem, "Graceful OSPF Restart", RFC 3623, DOI 10.17487/RFC3623, November 2003, <https://www.rfc-editor.org/info/rfc3623>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <https://www.rfc-editor.org/info/rfc4552>.

[RFC4576] Rosen, E., Psenak, P., and P. Pillay-Esnault, "Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4576, DOI 10.17487/RFC4576, June 2006, <https://www.rfc-editor.org/info/rfc4576>.

[RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, DOI 10.17487/RFC4577, June 2006, <https://www.rfc-editor.org/info/rfc4577>.

[RFC4750] Joyal, D., Ed., Galecki, P., Ed., Giacalone, S., Ed., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, DOI 10.17487/RFC4750, December 2006, <https://www.rfc-editor.org/info/rfc4750>.

[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>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <https://www.rfc-editor.org/info/rfc5082>.

[RFC5185] Mirtorabi, S., Psenak, P., Lindem, A., Ed., and A. Oswal, "OSPF Multi-Area Adjacency", RFC 5185, DOI 10.17487/RFC5185, May 2008, <https://www.rfc-editor.org/info/rfc5185>.

[RFC5187] Pillay-Esnault, P. and A. Lindem, "OSPFv3 Graceful Restart", RFC 5187, DOI 10.17487/RFC5187, June 2008, <https://www.rfc-editor.org/info/rfc5187>.

[RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, DOI 10.17487/RFC5250, July 2008, <https://www.rfc-editor.org/info/rfc5250>.

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[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>.

[RFC5613] Zinin, A., Roy, A., Nguyen, L., Friedman, B., and D. Yeung, "OSPF Link-Local Signaling", RFC 5613, DOI 10.17487/RFC5613, August 2009, <https://www.rfc-editor.org/info/rfc5613>.

[RFC5643] Joyal, D., Ed. and V. Manral, Ed., "Management Information Base for OSPFv3", RFC 5643, DOI 10.17487/RFC5643, August 2009, <https://www.rfc-editor.org/info/rfc5643>.

[RFC5838] Lindem, A., Ed., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, DOI 10.17487/RFC5838, April 2010, <https://www.rfc-editor.org/info/rfc5838>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[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>.

[RFC6565] Pillay-Esnault, P., Moyer, P., Doyle, J., Ertekin, E., and M. Lundberg, "OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol", RFC 6565, DOI 10.17487/RFC6565, June 2012, <https://www.rfc-editor.org/info/rfc6565>.

[RFC6860] Yang, Y., Retana, A., and A. Roy, "Hiding Transit-Only Networks in OSPF", RFC 6860, DOI 10.17487/RFC6860, January 2013, <https://www.rfc-editor.org/info/rfc6860>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <https://www.rfc-editor.org/info/rfc7490>.

[RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", RFC 7684, DOI 10.17487/RFC7684, November 2015, <https://www.rfc-editor.org/info/rfc7684>.

[RFC7770] Lindem, A., Ed., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 7770, DOI 10.17487/RFC7770, February 2016, <https://www.rfc-editor.org/info/rfc7770>.

[RFC7777] Hegde, S., Shakir, R., Smirnov, A., Li, Z., and B. Decraene, "Advertising Node Administrative Tags in OSPF", RFC 7777, DOI 10.17487/RFC7777, March 2016, <https://www.rfc-editor.org/info/rfc7777>.

[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>.

[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>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[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>.

[RFC8405] Decraene, B., Litkowski, S., Gredler, H., Lindem, A., Francois, P., and C. Bowers, "Shortest Path First (SPF) Back-Off Delay Algorithm for Link-State IGPs", RFC 8405, DOI 10.17487/RFC8405, June 2018, <https://www.rfc-editor.org/info/rfc8405>.

[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>.

7.2. Informative References

[RFC0905] "ISO Transport Protocol specification ISO DP 8073", RFC 905, DOI 10.17487/RFC0905, April 1984, <https://www.rfc-editor.org/info/rfc905>.

[RFC1765] Moy, J., "OSPF Database Overflow", RFC 1765, DOI 10.17487/RFC1765, March 1995, <https://www.rfc-editor.org/info/rfc1765>.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <https://www.rfc-editor.org/info/rfc5443>.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, DOI 10.17487/RFC5714, January 2010, <https://www.rfc-editor.org/info/rfc5714>.

[RFC6987] Retana, A., Nguyen, L., Zinin, A., White, R., and D. McPherson, "OSPF Stub Router Advertisement", RFC 6987, DOI 10.17487/RFC6987, September 2013, <https://www.rfc-editor.org/info/rfc6987>.

Appendix A. Contributors' Addreses

Dean Bogdanovic
Volta Networks, Inc.

EMail: dean@voltanet.io

Kiran Koushik Agrahara Sreenivasa
Cisco Systems
12515 Research Blvd, Bldg 4
Austin, TX 78681
USA

EMail: kkoushik@cisco.com

Authors' Addresses

Derek Yeung
Arrcus

EMail: derek@arrcus.com

Yingzhen Qu
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: yingzhen.qu@huawei.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: zzhang@juniper.net

Ing-Wher Chen
MITRE Corporation

EMail: ingwherchen@mitre.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513

EMail: acee@cisco.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 30, 2018

T. Li
Arista Networks
P. Psenak
Cisco Systems, Inc.

June 28, 2018

Dynamic Flooding on Dense Graphs
draft-li-dynamic-flooding-05

Abstract

Routing with link state protocols in dense network topologies can result in sub-optimal convergence times due to the overhead associated with flooding. This can be addressed by decreasing the flooding topology so that it is less dense.

This document discusses the problem in some depth and an architectural solution. Specific protocol changes for IS-IS, OSPFv2, and OSPFv3 are described in this document.

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 30, 2018.

Copyright Notice

Table of Contents

1. Introduction

In recent years, there has been increased focused on how to address the dynamic routing of networks that have a bipartite (a.k.a. spine-leaf or leaf-spine), Clos [Clos], or Fat Tree [Leiserson] topology. Conventional Interior Gateway Protocols (IGPs, i.e., IS-IS [ISO10589], OSPFv2 [RFC2328], and OSPFv3 [RFC5340]) under-perform, redundantly flooding information throughout the dense topology, leading to overloaded control plane inputs and thereby creating operational issues. For practical considerations, network architects have resorted to applying unconventional techniques to address the problem, applying BGP in the data center [RFC7938]. However it is very clear that using an Exterior Gateway Protocol as an IGP is sub-optimal, if only due to the configuration overhead.

The primary issue that is demonstrated when conventional mechanisms are applied is the poor reaction of the network to topology changes. Normal link state routing protocols rely on a flooding algorithm for state distribution. In a dense topology, this flooding algorithm is highly redundant, resulting in unnecessary overhead. Each node in the topology receives each link state update multiple times. Ultimately, all of the redundant copies will be discarded, but only after they have reached the control plane and been processed. This creates issues because significant link state database updates can become queued behind many redundant copies of another update. This delays convergence as the link state database does not stabilize promptly.

In a real world implementation, the packet queues leading to the control plane are necessarily of finite size, so if the flooding rate exceeds the update processing rate for long enough, the control plane will be obligated to drop incoming updates. If these lost updates are of significance, this will further delay stabilization of the link state database and the convergence of the network.

This is not a new problem. Historically, when routing protocols have been deployed in networks where the underlying topology is a complete graph, there have been similar issues. This was more common when the underlying link layer fabric presented the network layer with a full mesh of virtual connections. This was addressed by reducing the

flooding topology through IS-IS Mesh Groups [RFC2973], but this approach requires careful configuration of the flooding topology.

Thus, the root problem is not limited to massively scalable data centers. It exists with any dense topology at scale.

This problem is not entirely surprising. Link state routing protocols were conceived when links were very expensive and topologies were sparse. The fact that those same designs are sub-optimal in a dense topology should not come as a huge surprise. The fundamental premise that was addressed by the original designs was an environment of extreme cost and scarcity. Technology has progressed to the point where links are cheap and common. This represents a complete reversal in the economic fundamentals of network engineering. The original designs are to be commended for continuing to provide correct operation to this point, and optimizations for operation in today’s environment are to be expected.

1.1. 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 RFC 2119 [RFC2119].

2. Problem Statement

In a dense topology, the flooding algorithm that is the heart of conventional link state routing protocols causes a great deal of redundant messaging. This is exacerbated by scale. While the protocol can survive this combination, the redundant messaging is unnecessary overhead and delays convergence. Thus, the problem is to provide routing in dense, scalable topologies with rapid convergence.

3. Solution Requirements

A solution to this problem must then meet the following requirements:

Requirement 1 Provide a dynamic routing solution. Reachability must be restored after any topology change.

Requirement 2 Provide a significant improvement in convergence.

Requirement 3 The solution should address a variety of dense topologies. Just addressing a complete bipartite topology such as K5,8 is insufficient. Multi-stage Clos topologies must also be addressed, as well as topologies that are slight variants. Addressing complete graphs is a good demonstration of generality.

Requirement 4 There must be no single point of failure. The loss of any link or node should not unduly hinder convergence.

Requirement 5 Dense topologies are subgraphs of much larger topologies. Operational efficiency requires that the dense subgraph not operate in a radically different manner than the remainder of the topology. While some operational differences are permissible, they should be minimized. Changes to nodes outside of the dense subgraph are not acceptable. These situations occur when massively scaled data centers are part of an overall larger wide-area network. Having a second protocol operating just on this subgraph would add much more complexity at the edge of the subgraph where the two protocols would have to inter-operate.

4. Dynamic Flooding

We have observed that the combination of the dense topology and flooding on the physical topology in a scalable network is sub-optimal. However, if we decouple the flooding topology from the physical topology and only flood on a greatly reduced portion of that topology, we can have efficient flooding and retain all of the resilience of existing protocols.

In this idea, the flooding topology is computed either centrally on an elected node or in a distributed manner on all nodes that are supporting Dynamic Flooding. If the flooding topology is computed centrally, it is encoded into and distributed as part of the normal link state database. We call this the centralized mode of operation. If the flooding topology is computed in a distributed fashion, we call this the distributed mode of operation. Nodes within the dense topology would only flood on the flooding topology. On links outside of the normal flooding topology, normal database synchronization mechanisms (i.e., OSPF database exchange, IS-IS CSNPs) would apply, but flooding would not. New link state information that arrives from outside of the flooding topology suggests that the sender has a different or no flooding topology information and that the link state update should be flooded on the flooding topology as well.

Since the flooding topology is computed prior to topology changes, it does not factor into the convergence time and can be done when the topology is stable. The speed of the computation and its distribution, in the case of a centralized mode, is not a significant issue.

If a node does not have any flooding topology information when it receives new link state information, it should flood according to

legacy flooding rules. This situation will occur when the dense topology is first established, but is unlikely to recur.

When centralized mode is used and if, during a transient, there are multiple flooding topologies being advertised, then nodes should flood link state updates on all of the flooding topologies. Each node should locally evaluate the election of the lead node for the dense subgraph and first flood on the topology of the lead node. The rationale behind this is straightforward: if there is a transient and there has been a recent change in the elected node, then propagating topology information promptly along the most likely flooding topology should be the priority.

During transients, it is possible that loops will form in the flooding topology. This is not problematic, as the legacy flooding rules would cause duplicate updates to be ignored. Similarly, during transients, it is possible that the forwarding topology may become disconnected. To address this, nodes can perform a database synchronization check anytime a link is added to or removed from the flooding topology.

4.1. Applicability

In a complete graph, this approach is appealing because it drastically decreases the flooding topology without the manual configuration of mesh groups. By controlling the diameter of the flooding topology, as well as the maximum degree node in the flooding topology, convergence time goals can be met and the stability of the control plane can be assured.

Similarly, in a massively scaled data center, where there are many opportunities for redundant flooding, this mechanism ensures that flooding is redundant, with each leaf and spine well connected, while ensuring that no update need make too many hops and that no node shares an undue portion of the flooding effort.

In a network where only a portion of the nodes support Dynamic Flooding, the remaining nodes will continue to perform universal flooding. This is not an issue for correctness, as no node can become isolated.

Flooding that is initiated within the flooding topology will remain within that flooding topology until it reaches a legacy node, which will resume legacy flooding. Legacy flooding will be bounded by the flooding topology, which can help limit the propagation of unnecessary flooding. Whether or not the network can remain stable in this condition is unknown and may be very dependent on the number and location of the nodes that support Dynamic Flooding.

4.2. Leader election

A single node within the dense topology is elected as an Area Leader.

A generalization of the mechanisms used in existing Designated Router (OSPF) or Designated Intermediate-System (IS-IS) elections suffices. The elected node is known as the Area Leader.

In the case of centralized mode, the Area Leader is responsible for computing and distributing the flooding topology. When a new node is elected and has distributed new flooding topology information, then the old node should withdraw its flooding topology information from the link state database. If the old node does not return to the topology in a timely manner, the new node may remove the old node's information from the link state database.

In the case of distributed mode, the distributed algorithm advertised by the Area Leader MUST be used by all routers that participate in Dynamic Flooding.

Not every router needs to be a candidate to be Area Leader within an area, as a single candidate is sufficient for correct operation. For redundancy, however, it is strongly RECOMMENDED that there be multiple candidates.

4.3. Computing the Flooding Topology

There is a great deal of flexibility in how the flooding topology may be computed. For resilience, it needs to at least contain a cycle of all nodes in the dense subgraph. However, additional links could be added to decrease the convergence time. The trade-off between the density of the flooding topology and the convergence time is a matter for further study. The exact algorithm for computing the flooding topology in the case of the centralized computation need not be standardized, as it is not an interoperability issue. Only the encoding of the result needs to be documented. In the case of distributed mode, all nodes in the IGP area need to use the same algorithm to compute the flooding topology. It is possible to use private algorithms to compute flooding topology, so long as all nodes in the IGP area use the same algorithm.

While the flooding topology should be a covering cycle, it need not be a Hamiltonian cycle where each node appears only once. In fact, in many relevant topologies this will not be possible e.g., K5,8. This is fortunate, as computing a Hamiltonian cycle is known to be NP-complete.

A simple algorithm to compute the topology for a complete bipartite graph is to simply select unvisited nodes on each side of the graph until both sides are completely visited. If the number of nodes on each side of the graph are unequal, then revisiting nodes on the less populated side of the graph will be inevitable. This algorithm can run in O(N) time, so is quite efficient.

While a simple cycle is adequate for correctness and resiliency, it may not be optimal for convergence. At scale, a cycle may have a diameter that is half the number of nodes in the graph. This could cause an undue delay in link state update propagation. Therefore it may be useful to have a bound on the diameter of the flooding topology. Introducing more links into the flooding topology would reduce the diameter, but at the trade-off of possibly adding redundant messaging. The optimal trade-off between convergence time and graph diameter is for further study.

Similarly, if additional redundancy is added to the flooding topology, specific nodes in that topology may end up with a very high degree. This could result in overloading the control plane of those nodes, resulting in poor convergence. Thus, it may be optimal to have an upper bound on the degree of nodes in the flooding topology. Again, the optimal trade-off between graph diameter, node degree, and convergence time, and topology computation time is for further study.

If the leader chooses to include a multi-node broadcast LAN segment as part of the flooding topology, all of the connectivity to that LAN segment should be included as well. Once updates are flooded onto the LAN, they will be received by every attached node.

4.4. Topologies on Complete Bipartite Graphs

Complete bipartite graph topologies have become popular for data center applications and are commonly called leaf-spine or spine-leaf topologies. In this section, we discuss some flooding topologies that are of particular interest in these networks.

4.4.1. A Minimal Flooding Topology

We define a Minimal Flooding Topology on a complete bipartite graph as one in which the topology is connected and each node has at least degree two. This is of interest because it guarantees that the flooding topology has no single points of failure.

In practice, this implies that every leaf node in the flooding topology will have a degree of two. As there are usually more leaves than spines, the degree of the spines will be higher, but the load on the individual spines can be evenly distributed.

This type of flooding topology is also of interest because it scales well. As the number of leaves increases, we can construct flooding topologies that perform well. Specifically, for n spines and m leaves, if m >= n(n/2-1), then there is a flooding topology that has a diameter of four.

4.4.2. Xia Topologies

We define a Xia Topology on a complete bipartite graph as one in which all spine nodes are bi-connected through leaves with degree two, but the remaining leaves all have degree one and are evenly distributed across the spines.

Constructively, we can create a Xia topology by iterating through the spines. Each spine can be connected to the next spine by selecting any unused leaf. Since leaves are connected to all spines, all leaves will have a connection to both the first and second spine and we can therefore choose any leaf without loss of generality. Continuing this iteration across all of the spines, selecting a new leaf at each iteration, will result in a path that connects all spines. Adding one more leaf between the last and first spine will produce a cycle of n spines and n leaves.

At this point, m-n leaves remain unconnected. These can be distributed evenly across the remaining spines, connected by a single link.

Xia topologies represent a compromise that trades off increased risk and decreased performance for lower flooding amplification. Xia topologies will have a larger diameter. For m spines, the diameter will be m + 2.

In a Xia topology, some leaves are singly connected. This represents a risk in that in some failures, convergence may be delayed. However, there may be some alternate behaviors that can be employed to mitigate these risks. If a leaf node sees that its single link on the flooding topology has failed, it can compensate by performing a database synchronization check with a different spine. Similarly, if a leaf determines that its connected spine on the flooding topology has failed, it can compensate by performing a database synchronization check with a different spine. In both of these cases, the synchronization check is intended to ameliorate any delays in link state propagation due to the fragmentation of the flooding topology.

The benefit of this topology is that flooding load is easily understood. Each node in the spine cycle will never receive an

update more than twice. For n leaves and m spines, a spine never transmits more than m/n updates.

4.4.3. Optimization

If two systems have multiple links between them, only one of the links should be part of the flooding topology. Moreover, symmetric selection of the link to use for flooding is not required.

4.5. Encoding the Flooding Topology

There are a variety of ways that the flooding topology could be encoded efficiently. If the topology was only a cycle, a simple list of the nodes in the topology would suffice. However, this is insufficiently flexible as it would require a slightly different encoding scheme as soon as a single additional link is added. Instead, we choose to encode the flooding topology as a set of intersecting paths, where each path is a set of connected edges.

Other encodings are certainly possible. We have attempted to make a useful trade off between simplicity, generality, and space.

4.6. Analysis of Topology Changes

In this section, we explicitly consider a variety of different topological failures in the network and how dynamic flooding should address them.

4.6.1. Link Addition

If a link is added to the topology, the protocol will form a normal adjacency on the link and update the appropriate link state advertisements for the routers on either end of the link. These link state updates will be flooded on the flooding topology.

In centralized mode, the Area Leader, upon receiving these updates, may choose to retain the existing flooding topology or may choose to modify the flooding topology. If it elects to change the flooding topology, it will update the flooding topology in the link state database and flood it using the new flooding topology.

In distributed mode, any change in the topology, including the link addition, should trigger the flooding topology recalculation. This is done to ensure that all nodes converge on the same flooding topology, regardless of the time of the calculation.

4.6.2. Node Addition

In centralized mode, if a node is added to the topology, then at least one link is also added to the topology. The paragraph above applies and the Area Leader will necessarily need to add the new node to the flooding topology.

In distributed mode, the addition of a node should trigger flooding topology recalculation.

Until the new node is incorporated into the flooding topology at least a single link towards the new node MUST be added to the flooding topology locally on all of its neighbors.

4.6.3. Link Failures Off the Flooding Topology

If a link that is not part of the flooding topology fails, then the adjoining routers will update their link state advertisements and flood them on the flooding topology. There is no need for changes to the flooding topology.

4.6.4. Failure of the Area Leader

The failure of the Area Leader can be detected by observing that it is disconnected from the area topology. In this case, the Area Leader election process is repeated and a new Area Leader is elected.

In the centralized mode, the new Area Leader will compute a new flooding topology and flood it using the new flooding topology.

As an optimization, applicable to centralized mode, the new Area Leader MAY compute a new flooding topology that has as much in common as possible with the old flooding topology. This will minimize the risk of over-flooding.

4.6.5. Failures on the Flooding Topology

If there is a failure on the flooding topology, the adjoining routers will update their link state advertisements and flood them. If the original flooding topology is bi-connected, the flooding topology should still be connected despite a single failure.

In centralized mode, the Area Leader will notice the change in the flooding topology, recompute the flooding topology, and flood it using the new flooding topology.

In distributed mode, all routers supporting dynamic flooding will notice the change in the flooding topology and recompute the new flooding topology.

4.6.6. Recovery from Multiple Failures

In the unlikely event of multiple failures on the flooding topology, it may become disconnected. The nodes that remain active on the edges of the flooding topology will recognize this, update their own link state advertisements and flood them on the remainder of the flooding topology. At this point, nodes will be able to compute that the flooding topology is partitioned.

Note that this is very different from partitioning the area itself. The area may remain connected and forwarding may still be effective.

When this condition is detected, the flooding topology can no longer be expected to deliver link state updates in a prompt manner. Nodes on the edges of the flooding topology should perform database synchronization on all links not on the flooding topology. Updates received from off of the flooding topology should be flooded on the remaining flooding topology. Any links that provide updates or require updates that are not part of the flooding topology should temporarily be added to the flooding topology. This should repair the current flooding topology, albeit in a sub-optimal manner.

In centrailzed mode, the Area Leader will also detect this condition, compute a new flooding topology, and flood it using the new flooding topology.

In distributed mode, all routers that actively participate in Dynamic Flooding will compute the new flooding topology.

5. Protocol Elements

5.1. IS-IS TLVs

The following TLVs are added to IS-IS:

1. A TLV that an IS may inject into its LSP to indicate its preference for becoming Area Leader.

2. A TLV to carry the list of system IDs that compromise the flooding topology for the area.

3. A TLV to carry the adjacency matrix for the flooding topology for the area.

5.1.1. IS-IS Area Leader Sub-TLV

The Area Leader Sub-TLV allows a system to:

1. Indicate its eligibility and priority for becoming Area Leader.

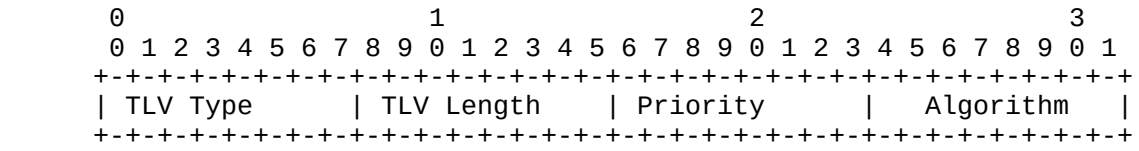
2. Indicate whether centralized or distributed mode is to be used to compute the flooding topology in the area.

3. Indicate the algorithm identifier for the algorithm that is used to compute the flooding topology in distributed mode.

Intermediate Systems (routers) that are not advertising this Sub-TLV are not eligible to become Area Leader.

The Area Leader is the router with the numerically highest Area Leader priority in the area. In the event of ties, the router with the numerically highest system ID is the Area Leader. Due to transients during database flooding, different routers may not agree on the Area Leader.

The Area Leader Sub-TLV is advertised as a Sub-TLV of the IS-IS Router Capability TLV-242 that is defined in [RFC7981] and has the following format:



TLV Type: TBD1

TLV Length: 2

Priority: 0-255, unsigned integer

Algorithm - a numeric identifier in the range 0-255 that identifies the algorithm used to calculate the flooding topology. The following values are defined:

0: Centralized computation by the Area Leader.

1-127: Standardized distributed algorithms. Individual values area assigned and managed by IANA. Before any assignments can be made, there MUST be an IETF specification that specifies IANA allocation for any value from this range (see Section 7.3).

128-254: Private distributed algorithms. Values from this range will not be registered with IANA and MUST NOT be mentioned by RFCs.

255: Reserved

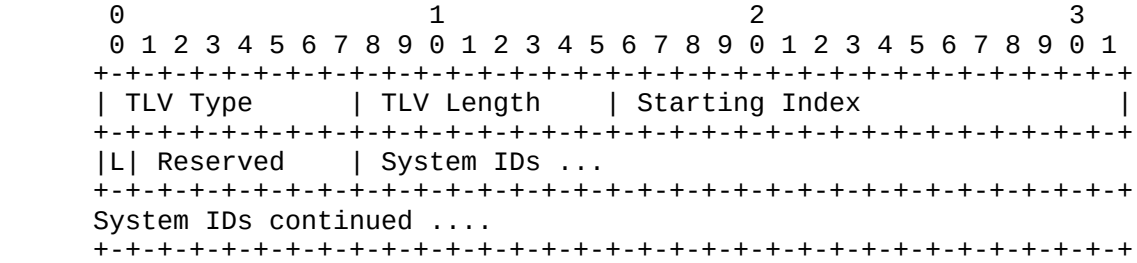
5.1.2. IS-IS Area System IDs TLV

IS-IS Area System IDs TLV is only used in centralized mode.

The Area System IDs TLV is used by the Area Leader to enumerate the system IDs that it has used in computing the flooding topology. Conceptually, the Area Leader creates a list of system IDs for all routers in the area, assigning indices to each system, starting with index 0.

Because the space in a single TLV is small, more than one TLV may be required to encode all of the system IDs in the area. This TLV may be present in multiple LSPs.

The format of the Area System IDs TLV is:



TLV Type: TBD2

TLV Length: 3 + (System ID length * (number of System IDs))

Starting index: The index of the first system ID that appears in this TLV.

L (Last): This bit is set if the index of the last system ID that appears in this TLV is equal to the last index in the full list of system IDs for the area.

System IDs: A concatenated list of system IDs for the area.

If there are multiple IS-IS Area System IDs TLVs with the L bit set advertised by the same router, the TLV which specifies the smaller maximum index is used and the other TLV(s) with L bit set are

ignored. TLVs which specify system IDs with indices greater than that specified by the TLV with the L bit set are also ignored.

5.1.3. IS-IS Flooding Path TLV

IS-IS Flooding Path TLV is only used in centralized mode.

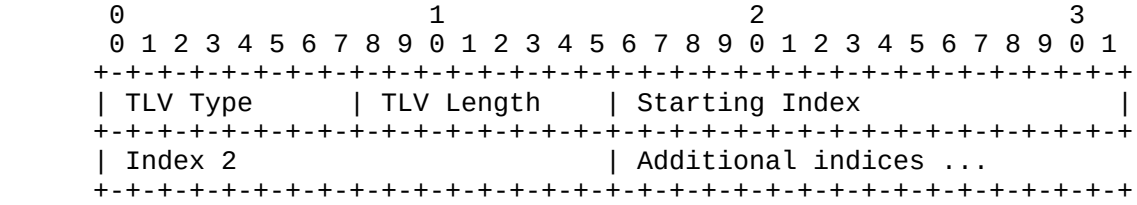
The Flooding Path TLV is used to denote a path in the flooding topology. The goal is an efficient encoding of the links of the topology. A single link is a simple case of a path that only covers two nodes. A connected path may be described as a sequence of indices: (I1, I2, I3, ...), denoting a link from the system with index 1 to the system with index 2, a link from the system with index 2 to the system with index 3, and so on.

If a path exceeds the size that can be stored in a single TLV, then the path may be distributed across multiple TLVs by the replication of a single system index.

Complex topologies that are not a single path can be described using multiple TLVs.

The Flooding Path TLV contains a list of system indices relative to the systems advertised through the Area System IDs TLV. At least 2 indices must be included in the TLV. Due to the length restriction of TLVs, this TLV can contain at most 126 system indices.

The Flooding Path TLV has the format:



TLV Type: TBD3

TLV Length: 2 * (number of indices in the path)

Starting index: The index of the first system in the path.

Index 2: The index of the next system in the path.

Additional indices (optional): A sequence of additional indices to systems along the path.

5.2. OSPF LSAs and TLVs

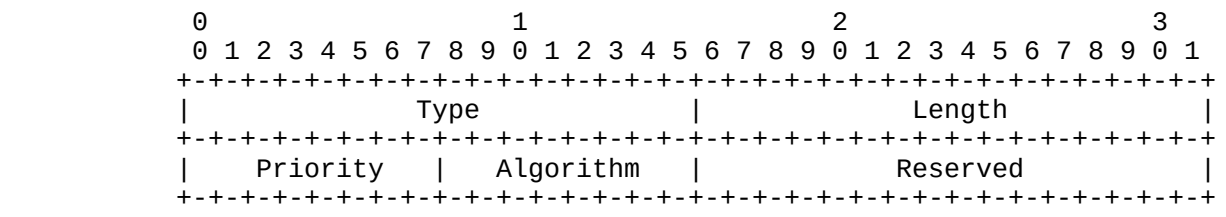
This section defines new LSAs and TLVs for both OSPFv2 and OSPFv3.

5.2.1. OSPF Area Leader Sub-TLV

The usage of the OSPF Area Leader Sub-TLV is identical to IS-IS and is described in Section 5.1.1.

The OSPF Area Leader Sub-TLV is used by both OSPFv2 and OSPFv3.

The OSPF Area Leader Sub-TLV is advertised as a top-level TLV of the RI LSA that is defined in [RFC7770] and has the following format:



Type: TBD4

Length: 4 octets

Priority: 0-255, unsigned integer

Algorithm: as defined in Section 5.1.1.

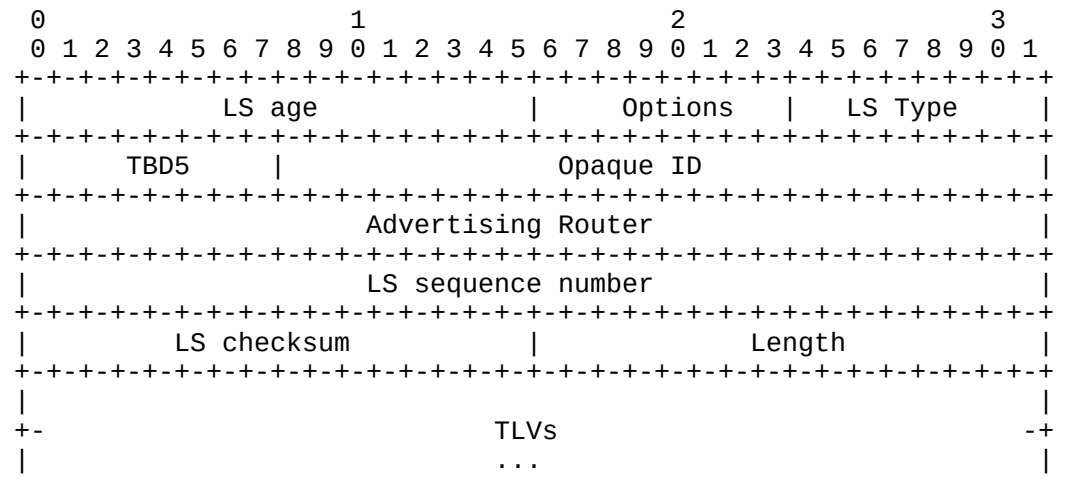
5.2.2. OSPFv2 Dynamic Flooding Opaque LSA

The OSPFv2 Dynamic Flooding Opaque LSA is only used in centralized mode.

The OSPFv2 Dynamic Flooding Opaque LSA is used to advertise additional data related to the dynamic flooding in OSPFv2. OSPFv2 Opaque LSAs are described in [RFC5250].

Multiple OSPFv2 Dynamic Flooding Opaque LSAs can be advertised by an OSPFv2 router. The flooding scope of the OSPFv2 Dynamic Flooding Opaque LSA is area-local.

The format of the OSPFv2 Dynamic Flooding Opaque LSA is as follows:



OSPFv2 Dynamic Flooding Opaque LSA

The opaque type used by OSPFv2 Dynamic Flooding Opaque LSA is TBD. The opaque type is used to differentiate the various type of OSPFv2 Opaque LSAs and is described in section 3 of [RFC5250]. The LS Type is 10. The LSA Length field [RFC2328] represents the total length (in octets) of the Opaque LSA including the LSA header and all TLVs (including padding).

The Opaque ID field is an arbitrary value used to maintain multiple Dynamic Flooding Opaque LSAs. For OSPFv2 Dynamic Flooding Opaque LSAs, the Opaque ID has no semantic significance other than to differentiate Dynamic Flooding Opaque LSAs originated by the same OSPFv2 router.

The format of the TLVs within the body of the OSPFv2 Dynamic Flooding Opaque LSA is the same as the format used by the Traffic Engineering Extensions to OSPF [RFC3630].

The Length field defines the length of the value portion in octets (thus a TLV with no value portion would have a length of 0). The TLV is padded to 4-octet alignment; padding is not included in the length field (so a 3-octet value would have a length of 3, but the total size of the TLV would be 8 octets). Nested TLVs are also 32-bit aligned. For example, a 1-octet value would have the length field set to 1, and 3 octets of padding would be added to the end of the value portion of the TLV. The padding is composed of zeros.

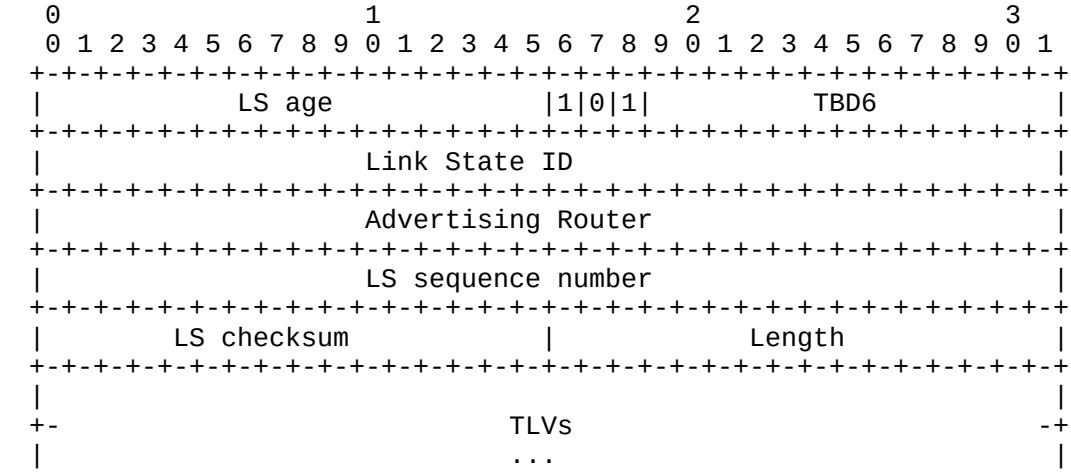
5.2.3. OSPFv3 Dynamic Flooding LSA

The OSPFv3 Dynamic Flooding Opaque LSA is only used in centralized mode.

The OSPFv3 Dynamic Flooding LSA is used to advertise additional data related to the dynamic flooding in OSPFv3.

The OSPFv3 Dynamic Flooding LSA has a function code of TBD. The flooding scope of the OSPFv3 Dynamic Flooding LSA is area-local. The U bit will be set indicating that the OSPFv3 Dynamic Flooding LSA should be flooded even if it is not understood. The Link State ID (LSID) value for this LSA is the Instance ID. OSPFv3 routers MAY advertise multiple Dynamic Flooding Opaque LSAs in each area.

The format of the OSPFv3 Dynamic Flooding LSA is as follows:



OSPFv3 Dynamic Flooding LSA

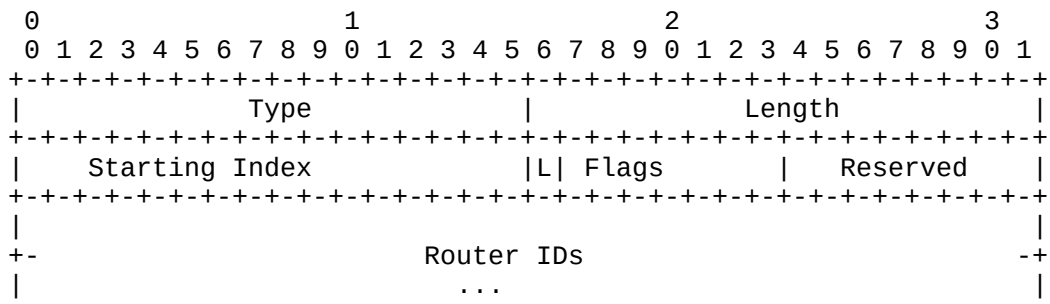
5.2.4. OSPF Area Router IDs TLV

The OSPF Area Router IDs TLV is a top level TLV of the OSPFv2 Dynamic Flooding Opaque LSA and OSPFv3 Dynamic Flooding LSA.

The OSPF Area Router IDs TLV is used by the Area Leader to enumerate the Router IDs that it has used in computing the flooding topology. Conceptually, the Area Leader creates a list of Router IDs for all routers in the area, assigning indices to each router, starting with index 0.

Because the space in a single OSPF Area Router IDs TLV is limited, more than one TLV may ve required to encode all of the Router IDs in the area. This TLV may also recur in multiple OSPFv2 Dynamic Flooding Opaque LSAs or OSPFv3 Dynamic Flooding LSA, so that all Router IDs can be advertised.

The format of the Area Router IDs TLV is:



OSPF Area Router IDs TLV

TLV Type: 1

TLV Length: 4 + (Router ID length * (number of Router IDs))

Starting index: The index of the first Router ID that appears in this TLV.

L (Last): This bit is set if the index of the last system ID that appears in this TLV is equal to the last index in the full list of Rourer IDs for the area.

Router IDs: A concatenated list of Router IDs for the area.

If there are multiple OSPF Area Router IDs TLVs with the L bit set advertised by the same router, the TLV which specifies the smaller maximum index is used and the other TLV(s) with L bit set are ignored. TLVs which specify Router IDs with indices greater than that specified by the TLV with the L bit set are also ignored.

5.2.5. OSPF Flooding Path TLV

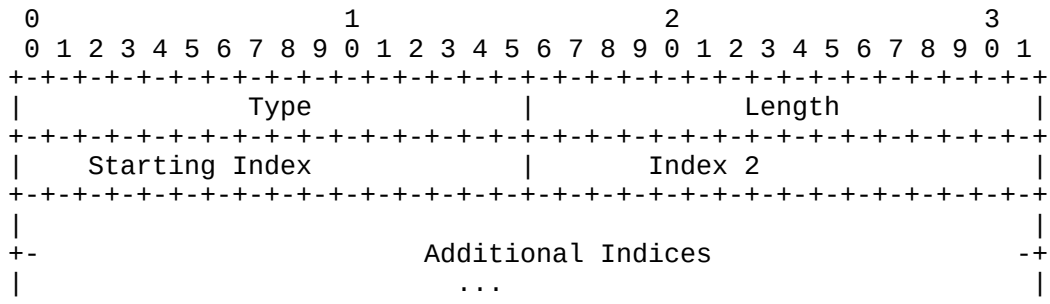
The OSPF Flooding Path TLV is a top level TLV of the OSPFv2 Dynamic Flooding Opaque LSAs and OSPFv3 Dynamic Flooding LSA.

The usage of the OSPF Flooding Path TLV is identical to IS-IS and is described in Section 5.1.3.

The OSPF Flooding Path TLV contains a list of Router ID indices relative to the Router IDs advertised through the OSPF Area Router IDs TLV. At least 2 indices must be included in the TLV.

Multiple OSPF Flooding Path TLVs can be advertised in a single OSPFv2 Dynamic Flooding Opaque LSA or OSPFv3 Dynamic Flooding LSA. OSPF Flooding Path TLVs can also be advertised in multiple OSPFv2 Dynamic Flooding Opaque LSAs or OSPFv3 Dynamic Flooding LSA, if they all can not fit in a single LSA.

The Flooding Path TLV has the format:



OSPF Flooding Path TLV

TLV Type: 2

TLV Length: 2 * (number of indices in the path)

Starting index: The index of the first Router ID in the path.

Index 2: The index of the next Router ID in the path.

Additional indices (optional): A sequence of additional indices to Router IDs along the path.

6. Behavioral Specification

In this section, we specify the detailed behaviors of the nodes participating in the IGP.

6.1. Leader Election

Any node that is capable MAY advertise its eligibility to become Area Leader.

Nodes that are not reachable are not eligible as Area Leader. Nodes that do not advertise their eligibility to become Area Leader are not eligible. Amongst the eligible nodes, the node with the numerically highest priority is the Area Leader. If multiple nodes all have the highest priority, then the node with the numerically highest system identifier in the case of IS-IS, or Router-ID in the case of OSPFv2 and OSPFv3 is the Area Leader.

6.2. Area Leader Responsibilities

If the Area Leader operates in centralized mode, it MUST advertise algorithm 0 in its Area Leader Sub-TLV. It also MUST compute and advertise a flooding topology for the area. The Area Leader MAY update the flooding topology at any time, however, it should not destabilize the network with undue or overly frequent topology changes.

The flooding topology MUST include all reachable nodes in the area. If nodes become unreachable on the flooding topology, the flooding topology MUST be recalculated. In centralized mode, the Area Leader MUST advertise a new flooding topology.

The flooding topology MAY be bi-connected. This is strongly RECOMMENDED but not required.

6.3. Distributed Flooding Topology Calculation

If the Area Leader advertises a non-zero algorithm in its Area Leader Sub-TLV, all routers in the area that support Dynamic Flooding and the value of algorithm advertised by the Area Leader MUST compute the flooding topology based on the Area Leader’s advertised algorithm. Routers that do not support the value of algorithm advertised by the Area Leader MUST continue to use legacy flooding mechanism as defined by the protocol.

If the value of the algorithm advertised by the Area Leader is from the range 128-254 (Private distributed algorithms), it is the responsibility of the network operator to guarantee that all nodes in the area have a common understanding of what the given algorithm value represents.

6.4. Flooding Behavior

Nodes that support Dynamic Flooding MUST use the flooding topology for flooding. The flooding topology is calculated locally in the case of distributed mode. In centralized mode the flooding topology is advertised in the area link state database. Link state updates received on one link in the flooding topology MUST be flooded on all other links in the flooding topology other than the link on which the update has been received. Link state updates received on a link not in the flooding topology MUST be flooded on all links in the flooding topology.

In centralized mode, if multiple flooding topologies are present in the area link state database, the node SHOULD flood on the union of the topologies.

When the flooding topology changes on a node, either as a result of the local computation in distributed mode or as a result of the advertisement from the Area Leader in centralized mode, the node MUST continue to flood on the union of the old and new flooding topology for a limited amount of time. This is required to provide all nodes sufficient time to migrate to the new flooding topology.

When failures occur, nodes will learn about them from link state updates and can compare those to the existing flooding topology. If the flooding topology becomes disconnected, then the nodes at the edges of the flooding topology should perform a database synchronization on all links. While the flooding topology is disconnected, if a new link state update is received on a link not in the flooding topology, then the node SHOULD temporarily consider the link as part of the flooding topology. When a new flooding topology is received or locally calculated, this MUST be discontinued.

7. IANA Considerations

7.1. IS-IS

This document requests the following code point from the "sub-TLVs for TLV 242" registry (IS-IS Router CAPABILITY TLV).

Type: TBD1

Description: IS-IS Area Leader Sub-TLV

Reference: This document (Section 5.1.1)

This document requests that IANA allocate and assign two code points from the "IS-IS TLV Codepoints" registry. One for each of the following TLVs:

Type: TBD2

Description: IS-IS Area System IDs TLV

Reference: This document (Section 5.1.2)

Type: TBD3

Description: IS-IS Flooding Path TLV

Reference: This document (Section 5.1.3)

7.2. OSPF

This document requests the following code point from the "OSPF Router Information (RI) TLVs" registry:

Type: TBD4

Description: OSPF Area Leader Sub-TLV

Reference: This document (Section 5.2.1)

This document requests the following code point from the "Opaque Link-State Advertisements (LSA) Option Types" registry:

Type: TBD5

Description: OSPFv2 Dynamic Flooding Opaque LSA

Reference: This document (Section 5.2.2)

This document requests the following code point from the "OSPFv3 LSA Function Codes" registry:

Type: TBD6

Description: OSPFv3 Dynamic Flooding LSA

Reference: This document (Section 5.2.3)

7.2.1. OSPF Dynamic Flooding LSA TLVs Registry

This specification also requests one new registry - "OSPF Dynamic Flooding LSA TLVs". New values can be allocated via IETF Review or IESG Approval

The "OSPF Dynamic Flooding LSA TLVs" registry will define top-level TLVs for the OSPFv2 Dynamic Flooding Opaque LSA and OSPFv3 Dynamic Flooding LSAs. It should be added to the "Open Shortest Path First (OSPF) Parameters" registries group.

The following initial values are allocated:

Type: 0

Description: Reserved

Reference: This document

Type: 1

Description: OSPF Area Router IDs TLV

Reference: This document (Section 5.2.4)

Type: 2

Description: OSPF Flooding Path TLV

Reference: This document (Section 5.2.5)

Types in the range 32768-33023 are for experimental use; these will not be registered with IANA, and MUST NOT be mentioned by RFCs.

Types in the range 33024-65535 are not to be assigned at this time. Before any assignments can be made in the 33024-65535 range, there MUST be an IETF specification that specifies IANA Considerations that covers the range being assigned.

7.3. IGP

IANA is requested to set up a registry called "IGP Algorithm Type For Computing Flooding Topology" under an existing "Interior Gateway Protocol (IGP) Parameters" IANA registries. The registration policy for this registry is "Standards Action" ([RFC8126] and [RFC7120]).

Values in this registry come from the range 0-255.

The initial values in the IGP Algorithm Type For Computing Flooding Topology registry are:

0: Reserved for centralized mode.

1-127: Available for standards action.

128-254: Reserved for private use.

255: Reserved.

8. Security Considerations

This document introduces no new security issues. Security of routing within a domain is already addressed as part of the routing protocols themselves. This document proposes no changes to those security architectures.

It is possible that an attacker could become Area Leader and introduce a flawed flooding algorithm into the network thus compromising the operation of the protocol. Authentication methods as describe in [RFC5304] and [RFC5310] for IS-IS, [RFC2328] and [RFC7474] for OSPFv2 and [RFC5340] and [RFC4552] for OSPFv3 SHOULD be used to prevent such attack.

9. Acknowledgements

The authors would like to thank Les Ginsberg, Zeqing (Fred) Xia, Naiming Shen, Adam Sweeney and Olufemi Komolafe for their helpful comments.

The authors would like to thank Tom Edsall for initially introducing them to the problem.

10. References

10.1. Normative References

[ISO10589] International Organization for Standardization, "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Nov. 2002.

[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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <https://www.rfc-editor.org/info/rfc4552>.

[RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, DOI 10.17487/RFC5250, July 2008, <https://www.rfc-editor.org/info/rfc5250>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[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>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <https://www.rfc-editor.org/info/rfc7474>.

[RFC7770] Lindem, A., Ed., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 7770, DOI 10.17487/RFC7770, February 2016, <https://www.rfc-editor.org/info/rfc7770>.

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <https://www.rfc-editor.org/info/rfc7981>.

[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>.

10.2. Informative References

[Clos] Clos, C., "A Study of Non-Blocking Switching Networks", The Bell System Technical Journal Vol. 32(2), DOI 10.1002/j.1538-7305.1953.tb01433.x, March 1953, <http://dx.doi.org/10.1002/j.1538-7305.1953.tb01433.x>.

[Leiserson] Leiserson, C., "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing", IEEE Transactions on Computers 34(10):892-901, 1985.

[RFC2973] Balay, R., Katz, D., and J. Parker, "IS-IS Mesh Groups", RFC 2973, DOI 10.17487/RFC2973, October 2000, <https://www.rfc-editor.org/info/rfc2973>.

[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>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

Authors' Addresses

Tony Li
Arista Networks
5453 Great America Parkway
Santa Clara, California 95054
USA

Email: tony.li@tony.li

Peter Psenak
Cisco Systems, Inc.
Eurovea Centre, Central 3
Pribinova Street 10
Bratislava 81109
Slovakia

Email: ppsenak@cisco.com

Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2018

N. Shen
L. Ginsberg
Cisco Systems
S. Thyamagundalu

June 18, 2018

IS-IS Routing for Spine-Leaf Topology
draft-shen-isis-spine-leaf-ext-06

Abstract

This document describes a mechanism for routers and switches in a Spine-Leaf type topology to have non-reciprocal Intermediate System to Intermediate System (IS-IS) routing relationships between the leafs and spines. The leaf nodes do not need to have the topology information of other nodes and exact prefixes in the network. This extension also has application in the Internet of Things (IoT).

Table of Contents

1. Introduction

The IS-IS routing protocol defined by [ISO10589] has been widely deployed in provider networks, data centers and enterprise campus environments. In the data center and enterprise switching networks, a Spine-Leaf topology is commonly used. This document describes a mechanism where IS-IS routing can be optimized for a Spine-Leaf topology.

In a Spine-Leaf topology, normally a leaf node connects to a number of spine nodes. Data traffic going from one leaf node to another leaf node needs to pass through one of the spine nodes. Also, the decision to choose one of the spine nodes is usually part of equal cost multi-path (ECMP) load sharing. The spine nodes can be considered as gateway devices to reach destinations on other leaf nodes. In this type of topology, the spine nodes have to know the topology and routing information of the entire network, but the leaf nodes only need to know how to reach the gateway devices to which are the spine nodes they are uplinked.

This document describes the IS-IS Spine-Leaf extension that allows the spine nodes to have all the topology and routing information, while keeping the leaf nodes free of topology information other than the default gateway routing information. The leaf nodes do not even need to run a Shortest Path First (SPF) calculation since they have no topology information.

1.1. 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 RFC 2119 [RFC2119].

2. Motivations

- The leaf nodes in a Spine-Leaf topology do not require complete topology and routing information of the entire domain since their forwarding decision is to use ECMP with spine nodes as default gateways

- The spine nodes in a Spine-Leaf topology are richly connected to leaf nodes, which introduces significant flooding duplication if they flood all Link State PDUs (LSPs) to all the leaf nodes. It saves both spine and leaf nodes' CPU and link bandwidth resources if flooding is blocked to leaf nodes. For small Top of the Rack (ToR) leaf switches in data centers, it is meaningful to prevent full topology routing information and massive database flooding through those devices.

- When a spine node advertises a topology change, every leaf node connected to it will flood the update to all the other spine nodes, and those spine nodes will further flood them to all the leaf nodes, causing a O(n^2) flooding storm which is largely redundant.

- Similar to some of the overlay technologies which are popular in data centers, the edge devices (leaf nodes) may not need to contain all the routing and forwarding information on the device's control and forwarding planes. "Conversational Learning" can be utilized to get the specific routing and forwarding information in the case of pure CLOS topology and in the events of link and node down.

- Small devices and appliances of Internet of Things (IoT) can be considered as leafs in the routing topology sense. They have CPU and memory constrains in design, and those IoT devices do not have to know the exact network topology and prefixes as long as there are ways to reach the cloud servers or other devices.

3. Spine-Leaf (SL) Extension

3.1. Topology Examples

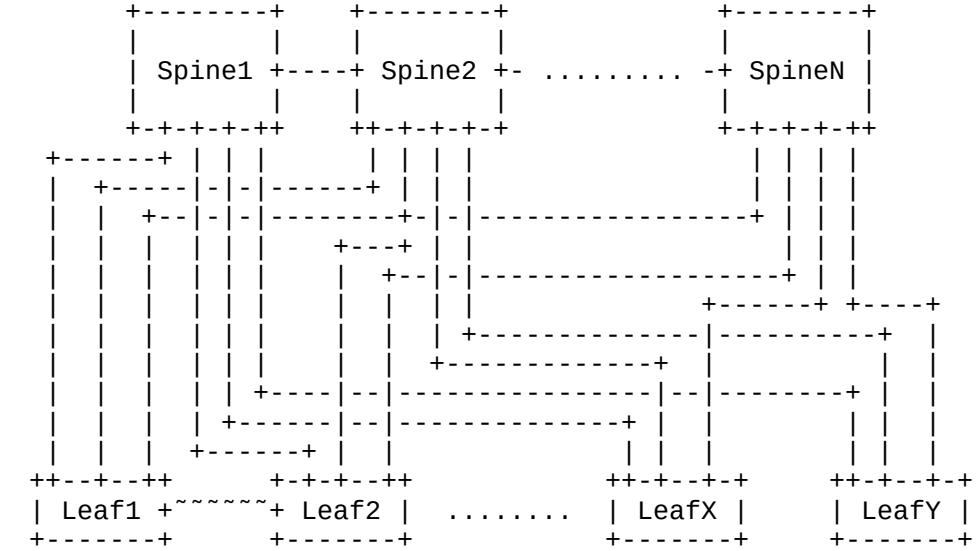


Figure 1: A Spine-Leaf Topology

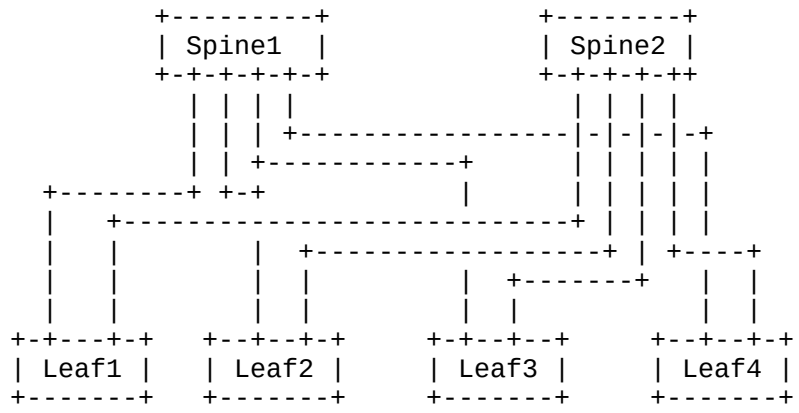


Figure 2: A CLOS Topology

3.2. Applicability Statement

This extension assumes the network is a Spine-Leaf topology, and it should not be applied in an arbitrary network setup. The spine nodes can be viewed as the aggregation layer of the network, and the leaf nodes as the access layer of the network. The leaf nodes use a load sharing algorithm with spine nodes as nexthops in routing and forwarding.

This extension works when the spine nodes are inter-connected, and it works with a pure CLOS or Fat Tree topology based network where the spines are NOT horizontally interconnected.

Although the example diagram in Figure 1 shows a fully meshed Spine-Leaf topology, this extension also works in the case where they are partially meshed. For instance, leaf1 through leaf10 may be fully meshed with spine1 through spine5 while leaf11 through leaf20 is fully meshed with spine4 through spine8, and all the spines are inter-connected in a redundant fashion.

This extension can also work in multi-level spine-leaf topology. The lower level spine node can be a 'leaf' node to the upper level spine node. A spine-leaf 'Tier' can be exchanged with IS-IS hello packets to allow tier X to be connected with tier X+1 using this extension. Normally tier-0 will be the TOR routers and switches if provisioned.

This extension also works with normal IS-IS routing in a topology with more than two layers of spine and leaf. For instance, in example diagrams Figure 1 and Figure 2, there can be another Core layer of routers/switches on top of the aggregation layer. From an IS-IS routing point of view, the Core nodes are not affected by this

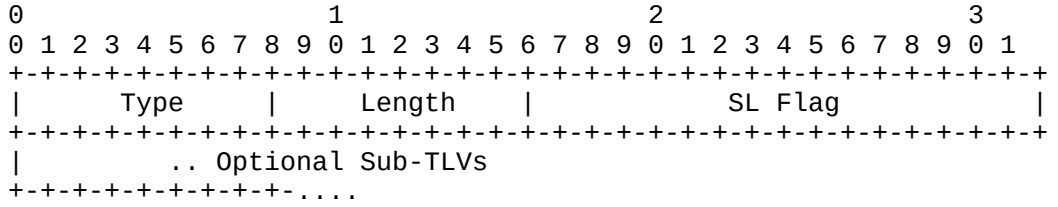
extension and will have the complete topology and routing information just like the spine nodes. To make the network even more scalable, the Core layer can operate as a level-2 IS-IS sub-domain while the Spine and Leaf layers operate as stays at the level-1 IS-IS domain.

This extension also supports the leaf nodes having local connections to other leaf nodes, in the example diagram Figure 1 there is a connection between 'Leaf1' node and 'Leaf2' node, and an external host can be dual homed into both of the leaf nodes.

This extension assumes the link between the spine and leaf nodes are point-to-point, or point-to-point over LAN [RFC5309]. The links connecting among the spine nodes or the links between the leaf nodes can be any type.

3.3. Spine-Leaf TLV

This extension introduces a new TLV, the Spine-Leaf TLV, which may be advertised in IS-IS Hello (IIH) PDUs, LSPs, or in Circuit Scoped Link State PDUs (CS-LSP) [RFC7356]. It is used by both spine and leaf nodes in this Spine-Leaf mechanism.

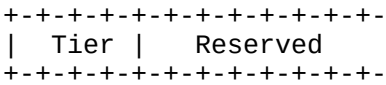


The fields of this TLV are defined as follows:

Type: 1 octet Suggested value 150 (to be assigned by IANA)

Length: 1 octet (2 + length of sub-TLVs).

SL Flags: 16 bits



Tier: A 4 bits value range from 0 to 15. It is used to represent the spine-leaf tier level when the 'T' bit is set. If the 'T' is cleared, this value MUST be set to zero from the sender, and it MUST be ignored on the receiver. The value 15 is reserved to indicate the tier level is unknown or not configured.

L bit (0x01): Only leaf node sets this bit. If the L bit is set in the SL flag, the node indicates it is in 'Leaf-Mode'.

R bit (0x02): Only Spine node sets this bit. If the R bit is set, the node indicates to the leaf neighbor that it can be used as the default route gateway.

B bit (0x04): Only leaf node sets this bit on Leaf-Leaf link, in additional to the 'L' bit setting. If the B bit is set, the node indicates to its leaf neighbor that it can be used as the backup default route gateway.

T bit (0x08): If set, the value in the 'Tier' field represents the spine-leaf tier level in the topology.

Optional Sub-TLV: Not defined in this document, for future extension

sub-TLVs MAY be included when the TLV is in a CS-LSP.
sub-TLVs MUST NOT be included when the TLV is in an IIH

3.3.1. Spine-Leaf Sub-TLVs

If the data center topology is a pure CLOS or Fat Tree, there are no link connections among the spine nodes. If we also assume there is not another Core layer on top of the aggregation layer, then the traffic from one leaf node to another may have a problem if there is a link outage between a spine node and a leaf node. For instance, in the diagram of Figure 2, if Leaf1 sends data traffic to Leaf3 through Spine1 node, and the Spine1-Leaf3 link is down, the data traffic will be dropped on the Spine1 node.

To address this issue spine and leaf nodes may send/request specific reachability information via the sub-TLVs defined below.

Two Spine-Leaf sub-TLVs are defined. The Leaf-Set sub-TLV and the Info-Req sub-TLV.

3.3.1.1. Leaf-Set Sub-TLV

This sub-TLV is used by spine nodes to optionally advertise Leaf neighbors to other Leaf nodes. The fields of this sub-TLV are defined as follows:

Type: 1 octet Suggested value 1 (to be assigned by IANA)

Length: 1 octet MUST be a multiple of 6 octets.

Leaf-Set: A list of IS-IS System-ID of the leaf node neighbors of this spine node.

3.3.1.2. Info-Req Sub-TLV

This sub-TLV is used by leaf nodes to request the advertisement of more specific prefix information from a selected spine node. The list of leaf nodes in this sub-TLV reflects the current set of leaf-nodes for which not all spine node neighbors have indicated the presence of connectivity in the Leaf-Set sub-TLV (See Section 3.3.1.1). The fields of this sub-TLV are defined as follows:

Type: 1 octet Suggested value 2 (to be assigned by IANA)

Length: 1 octet. It MUST be a multiple of 6 octets.

Info-Req: List of IS-IS System-IDs of leaf nodes for which connectivity information is being requested.

3.3.2. Advertising IPv4/IPv6 Reachability

In cases where connectivity between a leaf node and a spine node is down, the leaf node MAY request reachability information from a spine node as described in Section 3.3.1.2. The spine node utilizes TLVs 135 [RFC5305] and TLVs 236 [RFC5308] to advertise this information. These TLVs MAY be included either in IIHs or CS-LSPs sent from the spine to the requesting leaf node. Sending such information in IIHs has limited scale - all reachability information MUST fit within a single IIH. It is therefore recommended that CS-LSPs be used.

3.3.3. Advertising Connection to RF-Leaf Node

For links between Spine and Leaf Nodes on which the Spine Node has set the R-bit and the Leaf node has set the L-bit in their respective Spine-Leaf TLVs, spine nodes may advertise the link with a bit in the "link-attribute" sub-TLV [RFC5029] to express this link is not used for LSP flooding. This information can be used by nodes computing a flooding topology e.g., [DYNAMIC-FLOODING], to exclude the RF-Leaf nodes from the computed flooding topology.

3.4. Mechanism

Leaf nodes in a spine-leaf application using this extension are provisioned with two attributes:

1)Tier level of 0. This indicates the node is a Leaf Node. The value 0 is advertised in the Tier field of Spine-Leaf TLV defined above.

2)Flooding reduction enabled/disabled. If flooding reduction is enabled the L-bit is set to one in the Spine-Leaf TLV defined above

A spine node does not need explicit configuration. Spine nodes can dynamically discover their tier level by computing the number of hops to a leaf node. Until a spine node determines its tier level it MUST advertise level 15 (unknown tier level) in the Spine-Leaf TLV defined above.

When a spine node receives an IIH which includes the Spine-Leaf TLV with Tier level 0 and ’L’ bit set, it labels the point-to-point interface and adjacency to be a ’Reduced Flooding Leaf-Peer (RF-Leaf)’. IIHs sent by a spine node on a link to an RF-Leaf include the Spine-Leaf TLV with the ’R’ bit set in the flags field. The ’R’ bit indicates to the RF-Leaf neighbor that the spine node can be used as a default routing nexthop.

There is no change to the IS-IS adjacency bring-up mechanism for Spine-Leaf peers.

A spine node blocks LSP flooding to RF-Leaf adjacencies, except for the LSP PDUs in which the IS-IS System-ID matches the System-ID of the RF-Leaf neighbor. This exception is needed since when the leaf node reboots, the spine node needs to forward to the leaf node non-purged LSPs from the RF-Leaf’s previous incarnation.

Leaf nodes will perform IS-IS LSP flooding as normal over all of its IS-IS adjacencies, but in the case of RF-Leafs only self-originated LSPs will exist in its LSP database.

Spine nodes will receive all the LSP PDUs in the network, including all the spine nodes and leaf nodes. It will perform Shortest Path First (SPF) as a normal IS-IS node does. There is no change to the route calculation and forwarding on the spine nodes.

RF-Leaf nodes do not have any LSP in the network except for its own. Therefore there is no need to perform SPF calculation on the RF-Leaf node. It only needs to download the default route with the nexthops of those Spine Neighbors which have the 'R' bit set in the Spine-Leaf TLV in IIH PDUs. IS-IS can perform equal cost or unequal cost load sharing while using the spine nodes as nexthops. The aggregated metric of the outbound interface and the 'Reverse Metric' [REVERSE-METRIC] can be used for this purpose.

3.4.1. Pure CLOS Topology

In a data center where the topology is pure CLOS or Fat Tree, there is no interconnection among the spine nodes, and there is not another Core layer above the aggregation layer with reachability to the leaf nodes. When flooding reduction to RF-Leafs is in use, if the link between a spine and a leaf goes down, there is then a possibility of black holing the data traffic in the network.

As in the diagram Figure 2, if the link Spine1-Leaf3 goes down, there needs to be a way for Leaf1, Leaf2 and Leaf4 to avoid the Spine1 if the destination of data traffic is to Leaf3 node.

In the above example, the Spine1 and Spine2 are provisioned to advertise the Leaf-Set sub-TLV of the Spine-Leaf TLV. Originally both Spines will advertise Leaf1 through Leaf4 as their Leaf-Set. When the Spine1-Leaf3 link is down, Spine1 will only have Leaf1, Leaf2 and Leaf4 in its Leaf-Set. This allows the other leaf nodes to know that Spine1 has lost connectivity to the leaf node of Leaf3.

Each RF-Leaf node can select another spine node to request for some prefix information associated with the lost leaf node. In this diagram of Figure 2, there are only two spine nodes (Spine-Leaf topology can have more than two spine nodes in general). Each RF-Leaf node can independently select a spine node for the leaf information. The RF-Leaf nodes will include the Info-Req sub-TLV in the Spine-Leaf TLV in hellos sent to the selected spine node, Spine2 in this case.

The spine node, upon receiving the request from one or more leaf nodes, will find the IPv6/IPv4 prefixes advertised by the leaf nodes listed in the Info-Req sub-TLV. The spine node will use the mechanism defined in Section 3.3.2 to advertise these prefixes to the RF-Leaf node. For instance, it will include the IPv4 loopback prefix

of leaf3 based on the policy configured or administrative tag attached to the prefixes. When the leaf nodes receive the more specific prefixes, they will install the advertised prefixes towards the other spine nodes (Spine2 in this example).

For instance in the data center overlay scenario, when any IP destination or MAC destination uses the leaf3's loopback as the tunnel nexthop, the overlay tunnel from leaf nodes will only select Spine2 as the gateway to reach leaf3 as long as the Spine1-Leaf3 link is still down.

This negative routing is only relevant between tier 0 and tier 1 spine-leaf levels in a multi-level spine-leaf topology when the reduced flooding extension is in use. Nodes in tiers 1 or greater have the full topology information.

3.5. Implementation and Operation

3.5.1. CSNP PDU

In Spine-Leaf extension, Complete Sequence Number PDU (CSNP) does not need to be transmitted over the Spine-Leaf link to an RF-Leaf. Some IS-IS implementations send periodic CSNPs after the initial adjacency bring-up over a point-to-point interface. There is no need for this optimization here since the RF-Leaf does not need to receive any other LSPs from the network, and the only LSPs transmitted across the Spine-Leaf link is the leaf node LSP.

Also in the graceful restart case[RFC5306], for the same reason, there is no need to send the CSNPs over the Spine-Leaf interface to an RF-Leaf. Spine nodes only need to set the SRMflag on the LSPs belonging to the RF-Leaf.

3.5.2. Leaf to Leaf connection

Leaf to leaf node links are useful in host redundancy cases in switching networks, and normally there is no flooding extensions are required in this case. Each leaf node will set tier level = 0 in the Spine-Leaf TLV included in hellos to leaf neighbors. LSP will be exchanged over this link. In the example diagram Figure 1, the Leaf1 will get Leaf2's LSP and Leaf2 will get Leaf1's LSP. They will install more specific routes towards each other using this local Leaf-Leaf link. SPF will be performed in this case just like when the entire network only involves with those two IS-IS nodes. This does not affect the normal Spine-Leaf mechanism they perform toward the spine nodes.

Besides the local leaf-to-leaf traffic, the leaf node can serve as a backup gateway for its leaf neighbor. It needs to remove the 'Overload-Bit' setting in its LSP, and it sets both the 'L' bit and the 'B' bit in the SL-flag with a high 'Reverse Metric' value.

3.5.3. Overload Bit

The leaf node SHOULD set the 'overload' bit on its LSP PDU, since if the spine nodes were to forward traffic not meant for the local node, the leaf node does not have the topology information to prevent a routing/forwarding loop.

3.5.4. Spine Node Hostname

This extension creates a non-reciprocal relationship between the spine node and leaf node. The spine node will receive leaf's LSP and will know the leaf's hostname, but the leaf does not have spine's LSP. This extension allows the Dynamic Hostname TLV [RFC5301] to be optionally included in spine's IIH PDU when sending to a 'Leaf-Peer'. This is useful in troubleshooting cases.

3.5.5. IS-IS Reverse Metric

This metric is part of the aggregated metric for leaf's default route installation with load sharing among the spine nodes. When a spine node is in 'overload' condition, it should use the IS-IS Reverse Metric TLV in IIH [REVERSE-METRIC] to set this metric to maximum to discourage the leaf using it as part of the loadsharing.

In some cases, certain spine nodes may have less bandwidth in link provisioning or in real-time condition, and it can use this metric to signal to the leaf nodes dynamically.

In other cases, such as when the spine node loses a link to a particular leaf node, although it can redirect the traffic to other spine nodes to reach that destination leaf node, but it MAY want to increase this metric value if the inter-spine connection becomes over utilized, or the latency becomes an issue.

In the leaf-leaf link as a backup gateway use case, the 'Reverse Metric' SHOULD always be set to very high value.

3.5.6. Spine-Leaf Traffic Engineering

Besides using the IS-IS Reverse Metric by the spine nodes to affect the traffic pattern for leaf default gateway towards multiple spine nodes, the IPv6/IPv4 Info-Advertise sub-TLVs can be selectively used by traffic engineering controllers to move data traffic around the

data center fabric to alleviate congestion and to reduce the latency of a certain class of traffic pairs. By injecting more specific leaf node prefixes, it will allow the spine nodes to attract more traffic on some underutilized links.

3.5.7. Other End-to-End Services

Losing the topology information will have an impact on some of the end-to-end network services, for instance, MPLS TE or end-to-end segment routing. Some other mechanisms such as those described in PCE [RFC4655] based solution may be used. In this Spine-Leaf extension, the role of the leaf node is not too much different from the multi-level IS-IS routing while the level-1 IS-IS nodes only have the default route information towards the node which has the Attach Bit (ATT) set, and the level-2 backbone does not have any topology information of the level-1 areas. The exact mechanism to enable certain end-to-end network services in Spine-Leaf network is outside the scope of this document.

3.5.8. Address Family and Topology

IPv6 Address families[RFC5308], Multi-Topology (MT)[RFC5120] and Multi-Instance (MI)[RFC8202] information is carried over the IIH PDU. Since the goal is to simplify the operation of IS-IS network, for the simplicity of this extension, the Spine-Leaf mechanism is applied the same way to all the address families, MTs and MIs.

3.5.9. Migration

For this extension to be deployed in existing networks, a simple migration scheme is needed. To support any leaf node in the network, all the involved spine nodes have to be upgraded first. So the first step is to migrate all the involved spine nodes to support this extension, then the leaf nodes can be enabled with ’Leaf-Mode’ one by one. No flag day is needed for the extension migration.

4. IANA Considerations

A new TLV codepoint is defined in this document and needs to be assigned by IANA from the "IS-IS TLV Codepoints" registry. It is referred to as the Spine-Leaf TLV and the suggested value is 150. This TLV is only to be optionally inserted either in the IIH PDU or in the Circuit Flooding Scoped LSP PDU. IANA is also requested to maintain the SL-flag bit values in this TLV, and 0x01, 0x02 and 0x04 bits are defined in this document.

Value	Name	IIH	LSP	SNP	Purge	CS-LSP
150	Spine-Leaf	y	y	n	n	y

This extension also proposes to have the Dynamic Hostname TLV, already assigned as code 137, to be allowed in IIH PDU.

Value	Name	IIH	LSP	SNP	Purge
137	Dynamic Name	y	y	n	y

Two new sub-TLVs are defined in this document and needs to be added assigned by IANA from the "IS-IS TLV Codepoints". They are referred to in this document as the Leaf-Set sub-TLV and the Info-Req sub-TLV. It is suggested to have the values 1 and 2 respectively.

This document also requests that IANA allocate from the registry of link-attribute bit values for sub-TLV 19 of TLV 22 (Extended IS reachability TLV). This new bit is referred to as the "Connect to RF-Leaf Node" bit.

Value	Name	Reference
0x3	Connect to RF-Leaf Node	This document

5. Security Considerations

Security concerns for IS-IS are addressed in [ISO10589], [RFC5304], [RFC5310], and [RFC7602]. This extension does not raise additional security issues.

6. Acknowledgments

TBD.

7. Document Change Log

7.1. Changes to draft-shen-isis-spine-leaf-ext-05.txt

- Submitted January 2018.
- Just a refresh.

7.2. Changes to draft-shen-isis-spine-leaf-ext-04.txt

- Submitted June 2017.

- Added the Tier level information to handle the multi-level spine-leaf topology using this extension.

7.3. Changes to draft-shen-isis-spine-leaf-ext-03.txt

- Submitted March 2017.
- Added the Spine-Leaf sub-TLVs to handle the case of data center pure CLOS topology and mechanism.
- Added the Spine-Leaf TLV and sub-TLVs can be optionally inserted in either IIH PDU or CS-LSP PDU.
- Allow use of prefix Reachability TLVs 135 and 236 in IIHs/CS-LSPs sent from spine to leaf.

7.4. Changes to draft-shen-isis-spine-leaf-ext-02.txt

- Submitted October 2016.
- Removed the 'Default Route Metric' field in the Spine-Leaf TLV and changed to using the IS-IS Reverse Metric in IIH.

7.5. Changes to draft-shen-isis-spine-leaf-ext-01.txt

- Submitted April 2016.
- No change. Refresh the draft version.

7.6. Changes to draft-shen-isis-spine-leaf-ext-00.txt

- Initial version of the draft is published in November 2015.

8. References

8.1. Normative References

[ISO10589] ISO "International Organization for Standardization", "Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473), ISO/IEC 10589:2002, Second Edition.", Nov 2002.

[REVERSE-METRIC]
Shen, N., Amante, S., and M. Abrahamsson, "IS-IS Routing with Reverse Metric", draft-ietf-isis-reverse-metric-07 (work in progress), 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>.

[RFC5029] Vasseur, JP. and S. Previdi, "Definition of an IS-IS Link Attribute Sub-TLV", RFC 5029, DOI 10.17487/RFC5029, September 2007, <https://www.rfc-editor.org/info/rfc5029>.

[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>.

[RFC5301] McPherson, D. and N. Shen, "Dynamic Hostname Exchange Mechanism for IS-IS", RFC 5301, DOI 10.17487/RFC5301, October 2008, <https://www.rfc-editor.org/info/rfc5301>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5306] Shand, M. and L. Ginsberg, "Restart Signaling for IS-IS", RFC 5306, DOI 10.17487/RFC5306, October 2008, <https://www.rfc-editor.org/info/rfc5306>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <https://www.rfc-editor.org/info/rfc7356>.

[RFC7602] Chunduri, U., Lu, W., Tian, A., and N. Shen, "IS-IS Extended Sequence Number TLV", RFC 7602, DOI 10.17487/RFC7602, July 2015, <https://www.rfc-editor.org/info/rfc7602>.

[RFC8202] Ginsberg, L., Previdi, S., and W. Henderickx, "IS-IS Multi-Instance", RFC 8202, DOI 10.17487/RFC8202, June 2017, <https://www.rfc-editor.org/info/rfc8202>.

8.2. Informative References

[DYNAMIC-FLOODING] Li, T., "Dynamic Flooding on Dense Graphs", draft-li-dynamic-flooding (work in progress), 2018.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC5309] Shen, N., Ed. and A. Zinin, Ed., "Point-to-Point Operation over LAN in Link State Routing Protocols", RFC 5309, DOI 10.17487/RFC5309, October 2008, <https://www.rfc-editor.org/info/rfc5309>.

Authors' Addresses

Naiming Shen
Cisco Systems
560 McCarthy Blvd.
Milpitas, CA 95035
US

Email: naiming@cisco.com

Les Ginsberg
Cisco Systems
821 Alder Drive
Milpitas, CA 95035
US

Email: ginsberg@cisco.com

Sanjay Thyamagundalu

Email: tsanjay@gmail.com

LSR Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2018

A. Wang
China Telecom

June 28, 2018

OSPF Extend for Inter-Area Topology Retrieval
draft-wang-lsr-ospf-inter-area-topology-ext-00

Abstract

This document describes method to transfer the source router id of inter-area prefixes for OSPFv2 [RFC2328] and OSPFv3 [RFC5340], which is needed in topology retrieval processing for inter-area scenario.

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 30, 2018.

Copyright Notice

Table of Contents

1. Introduction

BGP-LS [RFC7752] describes the methodology that using BGP protocol to transfer the Link-State information. Such method can enable SDN controller to collect the underlay network topology automatically.

But if the underlay network is divided into multi area and running OSPF protocol, it is not easy for the SDN controller to rebuild the multi-area topology, because normally the ABR that locates on the boundary of different area will hide the detail topology information in non-backbone area, and the router in backbone area that runs BGP-LS protocol can only get and report the summary network information in non-backbone area.

[RFC7794] introduces "IPv4/IPv6 Source Router IDs" TLV to label the source of the prefixes redistributed from different Level, this TLV can be used to reconstruct the detail overall topology within level 1 and level 2. Such solution can also be applied into network that run OSPF protocol, but the related LSP message must be redefined.

This draft gives such solution for the OSPF v2 and OSPF v3 protocol.

2. 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] .

3. Inter-Area Topology Retrieval Scenario

Fig.1 illustrates the topology retrieval scenario when OSPF is running in multi-area. R0-R4 are routers in backbone area, S1-S4,T1-T4 are interal router in area 1 and area 2 respectively. R1 and R3 are border routers between area 0 and area 1; R2 and R4 are border routers between area 0 and area 2. N1 is the network between router S1 and S2, N2 is the network between router T1 and T2.

Normally, ABR router R1 or R3 will send the summary LSA(for OSPFv2) or Inter-Area-Prefix-LSAs(for OSPFv3) for network N1. When R0 receives such LSA, it can only know network N1 locates behind R1, and does not know where it is originated. When R0 reports the summary LSA information via BGP-LS protocol, the IP SDN controller can't certainly deduce the detail network topology within area 1. The situation is same as that in Area 2.

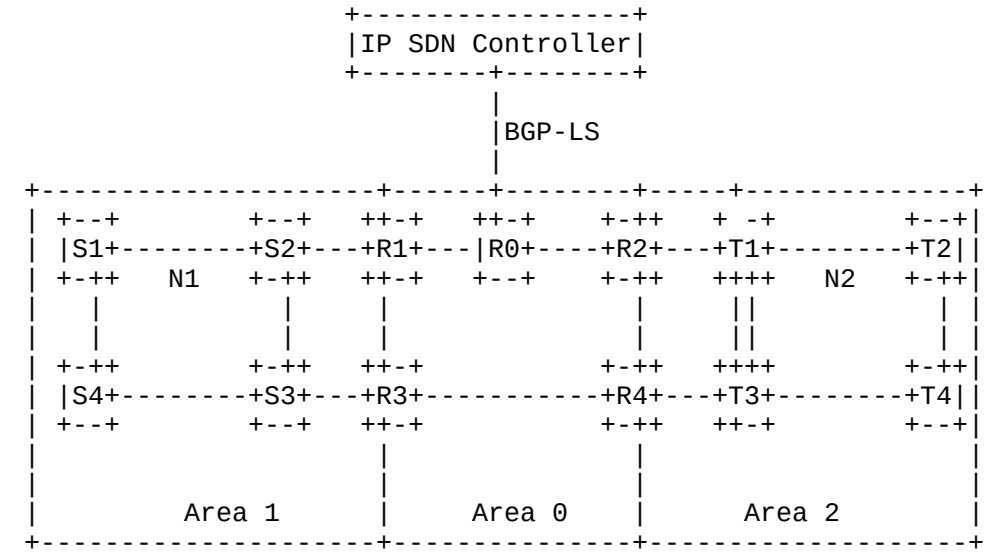


Fig.1 OSPF Inter-Area Topology Retrieval Scenario

If R0 has some methods to know the originator of network N1 and reports such information to IP SDN controller, then it is easy for the controller to retrieval the detail topology in non-backbone area.

Because traditional OSPFv2/v3 packet is not in the TLV format, we need to find some solutions to reuse or redefine the existing fields in summary LSA (OSPFv2) and Inter-Area-Prefix-LSAs(for OSPFv3)to transfer the additional information. The extend methods should not conflict with the usage of existing semantics.

Section 3.1 and section 3.2 give the proposed solutions for OSPFv2 and OSPFv3 respectively.

3.1. OSPFv2 Extend Solution (IPv4 Source Router ID)

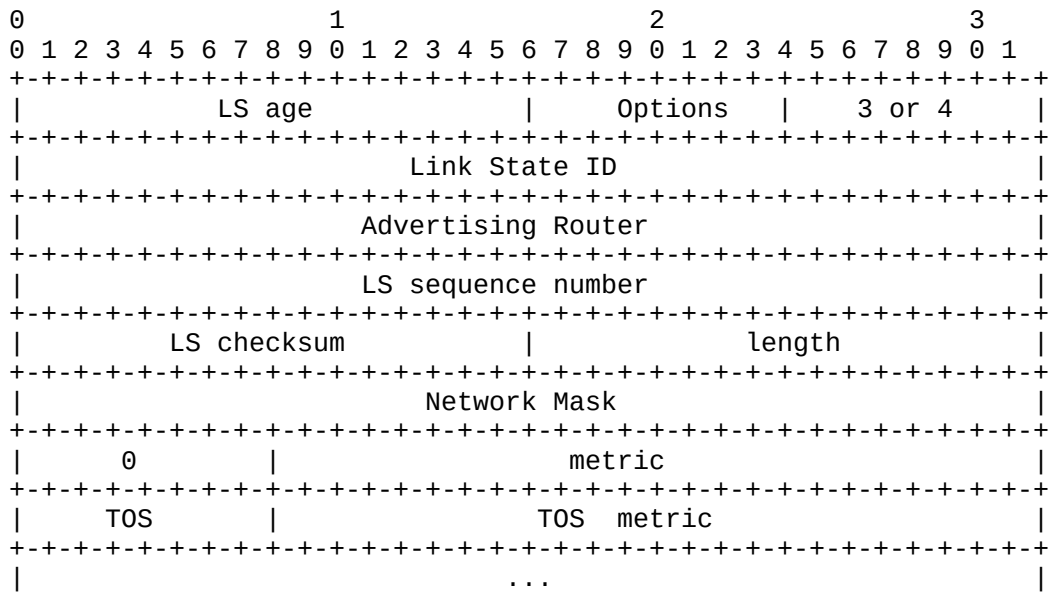


Fig.2 Summary LSA Format

Fig.2 illustrates the format of summary LSA. There is one byte that originately defined for the number of TOS types but in actually this feature does not applied in real network or implemented in the main stream router.

To transfer the additional information, this draft proposes to reuse/ redefine this field. In order to prevent possible conflict, even it is in very rare event, we can start the usage of this field from the upper limit, for example, 0xFE. Then the proposed extend summary LSA format is the followings:

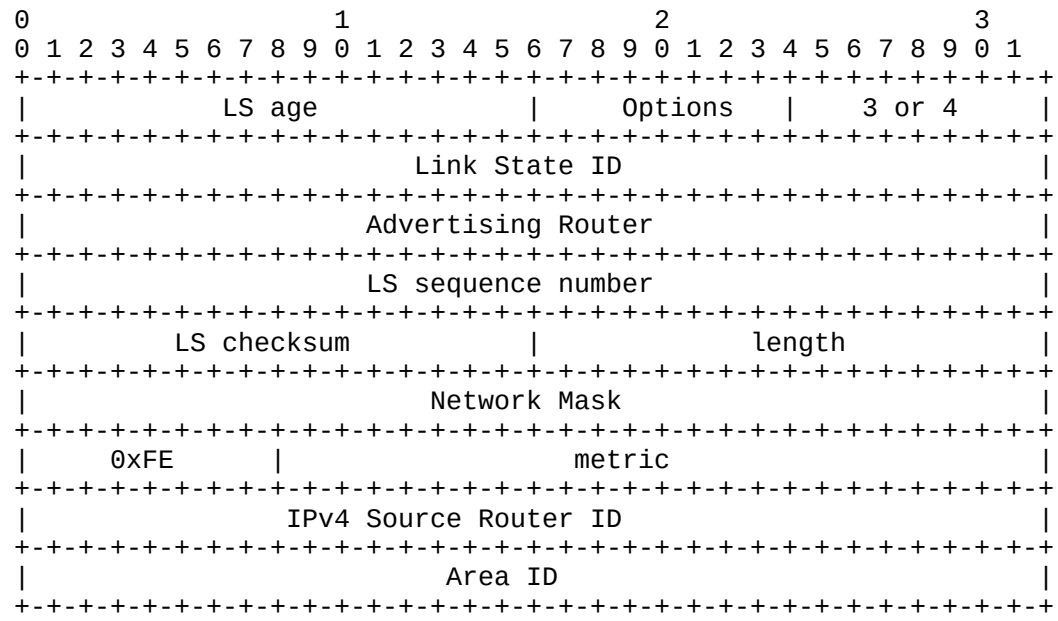


Fig.3 Extended Summary LSA Format

That is to say, if the field of "Numbers of TOS" equal "0xFE", then the "IPv4 Source Router ID"(4 bytes) of the inter-area network reported in summary LSA and its associated area id(4 bytes) are included in the field that follows the "metric" field.

3.2. OSPFv3 Extend Solution (IPv6 Source Router ID)

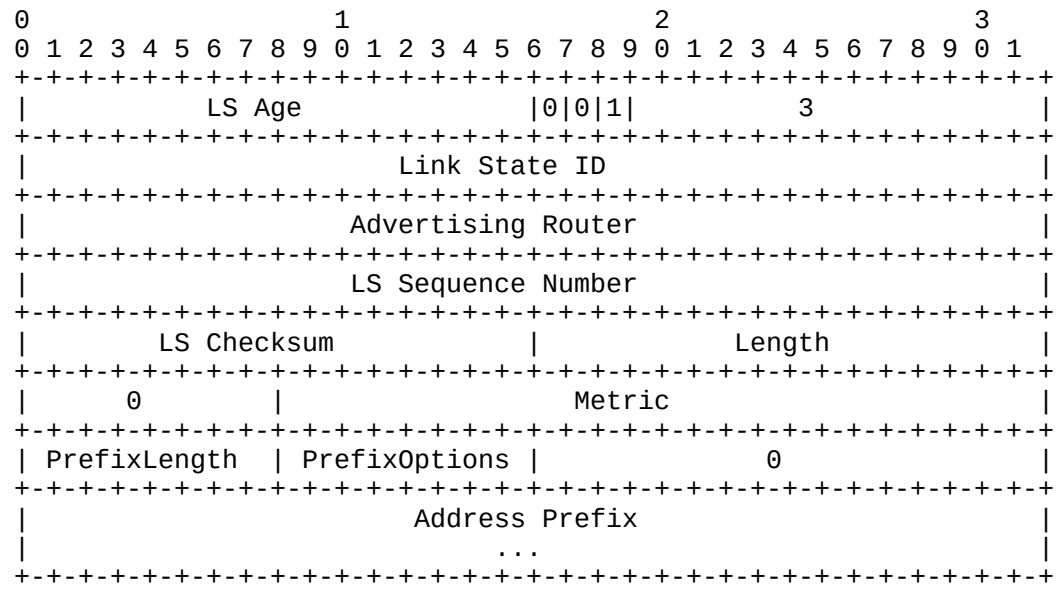


Fig.4 Inter-Area-Prefix-LSA Format

For OSPFv3, this draft proposes the similar method, because the semantic of the Inter-Area-Prefix-LSA format is almost same as the summary LSA format.

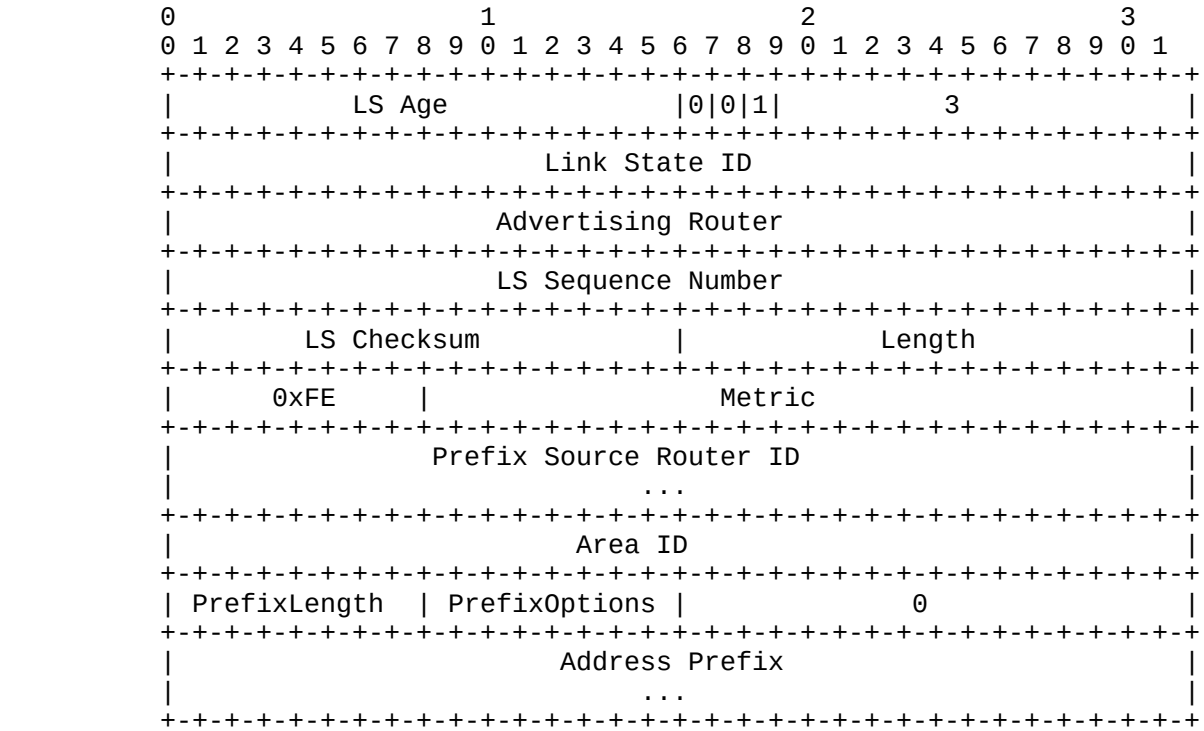


Fig.5 Extended Inter-Area-Prefix-LSA Format

If the value of "Numbers of TOS" equal "0xFE", then the "IPv6 source router ID" (16 bytes) and its corresponding area ID (4 bytes) information are inserted in the "Inter-Area-Prefix-LSA" after the field "Metric". After this, the normal Prefix information is followed as shown in Fig.5

3.3. Prefix Source Router ID sub TLV

[RFC7684] and [RFC8362] define the TLV format extension for OSPFv2 and OSPFv3 respectively. These documents give the flexibility to add new attributes for the prefixes and links. Based on these formats, we can define new sub TLV to transfer the "Prefix Source Router ID", as that defined in [RFC7794].

The proposed "Prefix Source Router ID" format is the following:

For IPv4 network, it is the following:

- Pv4 Source Router ID Type: TBD
- Length: 4

- o Value: IPv4 Router ID of the source of the advertisement

This sub TLV should be included in the "OSPFv2 Extended Prefix Opaque LSA" that defined in [RFC7684]

For IPv6 network, it is the following:

- o IPv6 Source Router ID Type: TBD
- o Length: 16
- o Value: IPv6 Router ID of the source of the advertisement

This sub TLV should be included in "E-Inter-Area-Prefix-LSA" that defined in [RFC8362]

3.4. Extend LSA generate process

When ABR(for example R1 in Fig.1)receives the "Router LSA" announcement in area 1, it should generate the corresponding extend "Summary LSA" or "Inter-Area-Prefix-LSA" that includes the "Source Router ID" of the network prefixes, which labels the corresponding link and the "area ID" that the source router belongs to.

When R0 receives such extend LSA, it then strips this additional information, put it into the corresponding part that in BGP-LS protocol as described in[I-D.wang-idr-bgpls-inter-as-topology-ext] and reports them to the IP SDN Controller.

3.5. Inter-Area Topology Retrieval Process

When IP SDN Controller receives this information, it should compare the prefix NLRI that included in the BGP-LS packet. When it encounters the same prefix but with different source router ID, it should extract the corresponding area ID, rebuild the link between these two different source router in non-backbone area.

Iterating the above process continuously, the IP SDN controller can then retrieve the detail topology that span multi-area.

4. Security Considerations

TBD.

5. IANA Considerations

TBD.

6. References

6.1. Normative References

[I-D.ietf-pce-pcep-extension-native-ip]
Wang, A., Khasanov, B., Cheruathur, S., and C. Zhu, "PCEP Extension for Native IP Network", draft-ietf-pce-pcep-extension-native-ip-00 (work in progress), June 2018.

[I-D.ietf-teas-native-ip-scenarios]
Wang, A., Huang, X., Qou, C., Huang, L., and K. Mi, "CCDR Scenario, Simulation and Suggestion", draft-ietf-teas-native-ip-scenarios-00 (work in progress), February 2018.

[I-D.ietf-teas-pcecc-use-cases]
Zhao, Q., Li, Z., Khasanov, B., Ke, Z., Fang, L., Zhou, C., Communications, T., and A. Rachitskiy, "The Use Cases for Using PCE as the Central Controller(PCECC) of LSPs", draft-ietf-teas-pcecc-use-cases-01 (work in progress), May 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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[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>.

[RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", RFC 7684, DOI 10.17487/RFC7684, November 2015, <https://www.rfc-editor.org/info/rfc7684>.

[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>.

[RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <https://www.rfc-editor.org/info/rfc7794>.

[RFC8362] Lindem, A., Roy, A., Goethals, D., Reddy Vallem, V., and F. Baker, "OSPFv3 Link State Advertisement (LSA) Extensibility", RFC 8362, DOI 10.17487/RFC8362, April 2018, <https://www.rfc-editor.org/info/rfc8362>.

6.2. Informative References

[I-D.wang-idr-bgpls-inter-as-topology-ext] Wang, A., "BGP-LS extend for inter-AS topology retrieval", draft-wang-idr-bgpls-inter-as-topology-ext-00 (work in progress), March 2018.

Author’s Address

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, Beijing 102209
China

Email: wangaj.bri@chinatelecom.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 17, 2018

R. White, Ed.
S. Zandi, Ed.
LinkedIn
June 15, 2018

IS-IS Support for Openfabric
draft-white-openfabric-06

Abstract

Spine and leaf topologies are widely used in hyperscale and cloud scale networks. In most of these networks, configuration is automated, but difficult, and topology information is extracted through broad based connections. Policy is often integrated into the control plane, as well, making configuration, management, and troubleshooting difficult. Openfabric is an adaptation of an existing, widely deployed link state protocol, Intermediate System to Intermediate System (IS-IS) that is designed to:

- Provide a full view of the topology from a single point in the network to simplify operations
- Minimize configuration of each Intermediate System (IS) (also called a router or switch) in the network
- Optimize the operation of IS-IS within a spine and leaf fabric to enable scaling

This document begins with an overview of openfabric, including a description of what may be removed from IS-IS to enable scaling. The document then describes an optimized adjacency formation process; an optimized flooding scheme; some thoughts on the operation of openfabric, metrics, and aggregation; and finally a description of the changes to the IS-IS protocol required for openfabric.

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 17, 2018.

Table of Contents

1. Introduction

1.1. Goals

Spine and leaf fabrics are often used in large scale data centers; in this application, they are commonly called a fabric because of their regular structure and predictable forwarding and convergence properties. This document describes modifications to the IS-IS protocol to enable it to run efficiently on a large scale spine and leaf fabric, openfabric. The goals of this control plane are:

- Provide a full view of the topology from a single point in the network to simplify operations
- Minimize configuration of each IS in the network
- Optimize the operation of IS-IS within a spine and leaf fabric to enable scaling

1.2. Contributors

The following people have contributed to this draft: Nikos Triantafillis (reflected flooding optimization), Ivan Pepelnjak (fabric locality calculation modifications), Christian Franke (fabric localigy calculation modification), Hannes Gredler (do not reflood optimizations), Les Ginsberg (capabilities encoding, circuit local reflooding), Naiming Shen (capabilities encoding, circuit local reflooding), Uma Chunduri (failure mode suggestions, flooding), Nick Russo, and Rodny Molina.

See [RFC5449], [RFC5614], and [RFC7182] for similar solutions in the Mobile Ad Hoc Networking (MANET) solution space.

1.3. Simplification

In building any scalable system, it is often best to begin by removing what is not needed. In this spirit, openfabric implementations MAY remove the following from IS-IS:

- External metrics. There is no need for external metrics in large scale spine and leaf fabrics; it is assumed that metrics will be properly configured by the operator to account for the correct order of route preference at any route redistribution point.
- Tags and traffic engineering processing. Openfabric is only designed to provide topology and reachability information. It is not designed to provide for traffic engineering, route preference through tags, or other policy mechanisms. It is assumed that all

routing policy will be provided through an overlay system which communicates directly with each IS in the fabric, such as PCEP [RFC5440] or I2RS [RFC7921]. Traffic engineering is assumed to be provided through Segment Routing (SR) [I-D.ietf-spring-segment-routing].

1.4. Additions and Requirements

To create a scalable link state fabric, openfabric includes the following:

- A slightly modified adjacency formation process.
- Mechanisms for determining which tier within a spine and leaf fabric in which the IS is located.
- A mechanism that reduces flooding to the minimum possible, while still ensuring complete database synchronization among the intermediate systems within the fabric.

Three general requirements are placed here; more specific requirements are considered in the following sections. Openfabric implementations:

- MUST support [RFC5301] and enable hostname advertisement by default if a hostname is configured on the intermediate system.
- SHOULD support [RFC6232], purge originator identification for IS-IS.
- MUST NOT be mixed with standard IS-IS implementations in operational deployments. Openfabric and standard IS-IS implementations SHOULD be treated as two separate protocols.

1.5. Sample Network

The following spine and leaf fabric will be used to describe these modifications.

Figure 1

To reduce confusion (spine and leaf fabrics are difficult to draw in plain text art), this diagram does not contain the connections between devices. The reader should assume that each device in a given layer is connected to every device in the layer above it. For instance:

- 5A is connected to 4A, 4B, 4C, 4D, 4E, and 4F
- 5B is connected to 4A, 4B, 4C, 4D, 4E, and 4F
- 4A is connected to 3A, 3B, 3C, 3D, 3E, 3F, 5A, 5B, 5C, 5D, 5E, and 5F
- 4B is connected to 3A, 3B, 3C, 3D, 3E, 3F, 5A, 5B, 5C, 5D, 5E, and 5F
- etc.

The tiers or stages of the fabric are also marked for easier reference. T0 is assumed to be connected to application servers, or rather they are Top of Rack (ToR) intermediate systems. The remaining tiers, T1 and T2, are connected only to the fabric itself. Note there are no "cross links," or "east west" links in the illustrated fabric. The fabric locality detection mechanism described here will not work if there are cross links running east/

west through the fabric. Locality detection may be possible in such a fabric; this is an area for further study.

2. Modified Adjacency Formation

Because Openfabric operates in a tightly controlled data center environment, various modifications can be made to the IS-IS neighbor formation process to increase efficencicy and simplify the protocol. Specifically, Openfabric implementations SHOULD support [RFC3719], section 4, hello padding for IS-IS. Variable hello padding SHOULD NOT be used, as data center fabrics are built using high speed links on which padded hellos will have little performance impact. Further modifications to the neighbor formation process are considered in the following sections.

2.1. Level 2 Adjacencies Only

Openfabric is designed to work in a single flooding domain over a single data center fabric at the scale of thousands of routers with hundreds of thousands of routes (so a moderate scale in router and route count terms). Because of the way Openfabric optimizes operation in this environment, it is not necessary nor desirable to build multiple flooding domains. For instance, the flooding optimizations described later this document require a full view of the topology, as does any proposed overlay to inject policy into the forwarding plane. In light of this, the following changes SHOULD BE to IS-IS implemetations to support Openfabric:

- IIH PDU 17 (level 2 point-to-point circuit hello) should be the only IIH PDU type transmitted (see section 9.7 of ISO 10589)
- In IIH PDU 17 (level 2 point-to-point circuit hello), the Circuit Type field should be set to 2 (see section 9.7 of ISO 10589)
- Support for IIH PDU 15 (level 1 broadcast hello) should be removed (see section 9.5 of ISO 10589)
- Support for IIH PDU 16 (level 2 broadcast hello) should be removed (see section 9.6 of ISO 10589)

2.2. Point-to-point Adjacencies

Data center network fabrics only contain point-to-point links; because of this, there is no reason to support any broadcast link types, nor to support the Designated Intermediate System processing, including pseudonode creation. In light ot his, processing related to sections 7.2.3 (broadcast networks), 7.3.8 (generation of level 1 pseudonode LSPs), 7.3.10 (generation of level 2 pseudonode LSPs), and

section 8.4.5 (LAN designated intermediate systems) in [ISO10589] SHOULD BE removed.

2.3. Three Way Handshake Support

It is important that two way connectivity be established before synchronizing the link state database, or routing through a link in a data center fabric. To reject optical failures that cause a one way connection between two routers, fabricDC must support the three way handshake mechanism described in [RFC5303].

2.4. Adjacency Formation Optimization

While adjacency formation is not considered particularly burdensome in IS-IS, it may still be useful to reduce the amount of state transferred across the network when connecting a new IS to the fabric. In its simplest form, the process is:

- An IS connected to the fabric will send hellos on all links.
- The IS will only complete the three-way handshake with one newly discovered neighbor; this would normally be the first neighbor which sends the newly connected intermediate system’s ID back in the three-way handshake process.
- The IS will complete its database exchange with this one newly adjacent neighbor.
- Once this process is completed, the IS will continue processing the remaining neighbors as normal.
- If synchronization is not achieved within twice the dead timer on the local interface, the newly connected IS will repeat this process with the second neighbor with which it forms a three-way adjacency.

This process allows each IS newly added to the fabric to exchange a full table once; a very minimal amount of information will be transferred with the remaining neighbors to reach full synchronization.

Any such optimization is bound to present a tradeoff between several factors; the mechanism described here increases the amount of time required to form adjacencies slightly in order to reduce the total state carried across the network. An alternative mechanism could provide a better balance of the amount of information carried across the network for initial synchronization and the time required to synchronize a new IS. For instance, an IS could choose to

synchronize its database with two or three adjacent intermediate systems, which could speed the synchronization process up at the cost of carrying additional data on the network. A locally determined balance between the speed of synchronization and the amount of data carried on the network can be acheived by adjusting the number of adjacent intermediate systems the newly attached IS synchronizes with.

3. Advertisement of Reachability Information

IS-IS describes the topology in two different sets of TLVs; the first describes the set of neighbors connected to an IS, the second describes the set of reachable destination connected to an IS. There are two different forms of both of these descriptions, one of which carries what are widely called narrow metrics, the other of which carries what are widely called wide metrics. In a tightly controlled data center fabric implementation, such as the ones Openfabric is designed to support, no IS that supports narrow metrics will ever be deployed or supported; hence there is no reason to support any metric type other than wide metrics.

- The Level 2 Link State PDU (type 20 in section 9.9 of [ISO10589]) and the scoped flooding PDU (type 10 in section 3.1 of [RFC7356]) SHOULD BE the only PDU types used to carry link state information in a Openfabric implementation

- Processing related to the Level 1 Link State PDU (type 18) MAY BE removed from Openfabric implementations (see section 9.8 of [ISO10589])

- Neighbor reachability MUST BE carried in TLV type 22 (see section 3 of [RFC5305])

- IPv4 reachability SHOULD BE carried in TLV type 135 (see section 4 of [RFC5305]), or TLV type 235 for multitopology implementations (see [RFC5120])

- IPv6 reachability SHOULD BE carried in TLV type 236 (see [RFC5308]), or TLV type 237 for multitopology implemenations (see [RFC5120])

- Processing related to the neighbor reachability TLV (type 2, see sections 9.8 and 9.9 of [ISO10589]) SHOULD BE removed

- Processing related to the narrow metric IP reachability TLV (types 128 and 130) SHOULD BE removed

Further, if segment routing support is desired, Openfabric MAY support the Prefix Segment Identifier sub-TLV and other TLVs as required in [I-D.ietf-isis-segment-routing-extensions].

4. Determining and Advertising Location on the Fabric

The tier to which a IS is connected is useful to enable autoconfiguration of intermediate systems connected to the fabric and to reduce flooding. Once the tier of an intermediate system within the fabric has been determined, it MUST be advertised using the 4 bit Tier field described in section 3.3 of [I-D.shen-isis-spine-leaf-ext]. This section describes a method of calculating the tier number, assuming the tier numbers rise in value from the edge of the fabric.

This method begins with two of the T0 intermediate systems advertising their location in the fabric. This information can either be obtained through:

- Two T0 intermediate systems are manually configured to advertise 0x00 in their IS reachability tier sub-TLV, indicating they are at the edge of the fabric (a ToR IS).

- The T0 intermediate systems detect they are T0 through the presence connected hosts (i.e. through a request for address assignment or some other means). If such detection is used, and the IS determines it is located at T0, it should advertise 0x00 in its IS reachability tier sub-TLV.

If the first method is used, the two T0 routers MUST be "maximally separated" on the fabric. They must be a maximal number of hops apart, or rather thay MUST NOT be connected to the same T1 device as their "upstream" towards the superspines in a 5 ary fabric.

The second method above SHOULD be used with care, as it may not be secure, and it may not work in all data center environments. For instance, if a host is mistakenly (or intentionally, as a form of attack) attached to a spine IS, or a request for address assignment is transmitted to a spine IS during the bootup phase of the device or fabric, it is possible to cause a spine IS to advertise itself as a T0. Unless the autodetection of the T0 devices is secured, the manual mechanism SHOULD BE used (configuring at least one T0 device manually).

Given the correct configuration of two T0 devices, maximally spaced on the fabric, the remaining intermediate systems calculate their tier number as follows:

- The local IS calculates an SPT (using SPF) setting the cost of every link to 1; this effectively calculates a topology only view of the network, without considering any configured link costs
- Ensure that at least two T0 are in the calculated SPT; otherwise abort
- Find the furthest T0; call this node A and set LD to the cost; the "farthest T0" is the T0 with the largest metric, or the farthest distance from the local calculating node
- Calculate an SPT (using SPF) from the perspective of A (above) setting the cost of every link to 1
- Find the furthest IS in A's SPT; call this node B and set RD to the cost from A to B
- Calculate the tier number of the local IS by subtracting LD from RD

In the example network, assume 5A and 1C are manually configured as a T0, and are advertising their tier numbers. From here:

- From 1A the path to 5A is 4 hops; this is LD
- Run SPF from the perspective of 5A with all link metrics set to 1
- From 5A the path length to 1C is 4; this is RD
- RD - LD is 0 at 1A, so 1A is T0, or a ToR

This process will work for any spine and leaf fabric without "cross links."

5. Flooding Optimization

Flooding is perhaps the most challenging scaling issue for a link state protocol running on a dense, large scale fabric. To reduce the flooding of link state information in the form of Link State Protocol Data Units (LSPs), Openfabric takes advantage of information already available in the link state protocol, the list of the local intermediate system's neighbor's neighbors, and the fabric locality computed above. The following tables are required to compute a set of reflooders:

- Neighbor List (NL) list: The set of neighbors

- Neighbor’s Neighbors (NN) list: The set of neighbor’s neighbors; this can be calculated by running SPF truncated to two hops

- Do Not Reflood (DNR) list: The set of neighbors who should have LSPs (or fragments) who should not reflood LSPs

- Reflood (RF) list: The set of neighbors who should flood LSPs (or fragments) to their adjacent neighbors to ensure synchronization

NL is set to contain all neighbors, and sorted deterministically (for instance, from the highest IS identifier to the lowest). All intermediate systems within a single fabric SHOULD use the same mechanism for sorting the NL list. NN is set to contain all neighbor’s neighbors, or all intermediate systems that are two hops away, as determined by performing a truncated SPF. The DNR and RF tables are initially empty. To begin, the following steps are taken to reduce the size of NN and NL:

- Move any IS in NL with its tier (or fabric location) set to T0 to DNR

- Remove all intermediate systems from NL and NN that in the shortest path to the IS that originated the LSP

Then, for every IS in NL:

- If the current entry in NL is connected to any entries in NN:

  * Move the IS to RF

  * Remove the intermediate systems connected to the IS from NN

- Else move the IS to DNR

When flooding, LSPs transmitted to adjacent neighbors on the RF list will be transmitted normally. Adjacent intermediate systems on this list will reflood received LSPs into the next stage of the topology, ensuring database synchronization. LSPs transmitted to adjacent neighbors on the DNR list, however, MUST be transmitted using a circuit scope PDU as described in [RFC7356].

5.1. Flooding Failures

It is possible in some failure modes for flooding to be incomplete because of the flooding optimizations outlined. Specifically, if a reflooder fails, or is somehow disconnected from all the links across which it should be reflooding, it is possible an LSP is only

partially flooded through the fabric. To prevent such situations, any IS receiving an LSP transmitted using DNR SHOULD:

- Set a short timer; the default should be less than one second
- When the timer expires, send a Complete Sequence Number Packet (CSNP) to all neighbors
- Process any Partial Sequence Number Packets (PSNPs) as required to resynchronize
- If a resynchronization is required, notify the network operator through a network management system

6. Other Optimizations

6.1. Transit Link Reachability

In order to reduce the amount of control plane state carried on large scale spine and leaf fabrics, openfabric implementations SHOULD NOT advertise reachability for transit links. These links MAY remain unnumbered, as IS-IS does not require layer 3 IP addresses to operate. Each IS SHOULD be configured with a single loopback address, which is assigned an IPv6 address, to provide reachability to intermediate systems which make up the fabric.

[RFC3277] SHOULD be supported on devices supporting openfabric with unnumbered interface in order to support traceability and network management.

6.2. Transiting T0 Intermediate Systems

In data center fabrics, ToR intermediate systems SHOULD NOT be used to transit between two T1 (or above) spine intermediate systems. The simplest way to prevent this is to set the overload bit [RFC3277] for all the LSPs originated from T0 intermediate systems. However, this solution would have the unfortunate side effect of causing all reachability beyond any T0 IS to have the same metric, and many implementations treat a set overload bit as a metric of 0xFFFF in calculating the Shortest Path Tree (SPT). This document proposes an alternate solution which preserves the leaf node metric, while still avoiding transiting T0 intermediate systems.

Specifically, all T0 intermediate systems SHOULD advertise their metric to reach any T1 adjacent neighbor with a cost of 0XFFE. T1 intermediate systems, on the other hand, will advertise T0 intermediate systems with the actual interface cost used to reach the T0 IS. Hence, links connecting T0 and T1 intermediate systems will

be advertised with an asymmetric cost that discourages transiting T0 intermediate systems, while leaving reachability to the destinations attached to T0 devices the same.

7. Openfabric and Route Aggregation

While schemes may be designed so reachability information can be aggregated in Openfabric deployments, this is not a recommended configuraiton.

8. Security Considerations

This document outlines modifications to the IS-IS protocol for operation on large scale data center fabrics. While it does add new TLVs, and some local processing changes, it does not add any new security vulnerabilities to the operation of IS-IS. However, openfabric implementations SHOULD implement IS-IS cryptographic authentication, as described in [RFC5304], and should enable other security measures in accordance with best common practices for the IS-IS protocol.

If T0 intermediate systems are auto-detected using information outside Openfabric, it is possible to attack the calucations used for flooding reduction and auto-configuration of intermediate systems. For instance, if a request for an address pool is used as an indicator of an attached host, and hence receiving such a request causes an intermediate system to advertise itself as T0, it is possible for an attacker (or a simple mistake) to cause auto-configuration to fail. Any such auto-detection mechanims SHOULD BE secured using appropriate techniques, as described by any protocols or mechanisms used.

9. References

9.1. Normative References

[I-D.shen-isis-spine-leaf-ext]
Shen, N., Ginsberg, L., and S. Thyamagundalu, "IS-IS Routing for Spine-Leaf Topology", draft-shen-isis-spine-leaf-ext-05 (work in progress), January 2018.

[ISO10589]
International Organization for Standardization, "Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Second Edition, Nov 2002.

[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>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <https://www.rfc-editor.org/info/rfc2629>.

[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>.

[RFC5301] McPherson, D. and N. Shen, "Dynamic Hostname Exchange Mechanism for IS-IS", RFC 5301, DOI 10.17487/RFC5301, October 2008, <https://www.rfc-editor.org/info/rfc5301>.

[RFC5303] Katz, D., Saluja, R., and D. Eastlake 3rd, "Three-Way Handshake for IS-IS Point-to-Point Adjacencies", RFC 5303, DOI 10.17487/RFC5303, October 2008, <https://www.rfc-editor.org/info/rfc5303>.

[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>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5309] Shen, N., Ed. and A. Zinin, Ed., "Point-to-Point Operation over LAN in Link State Routing Protocols", RFC 5309, DOI 10.17487/RFC5309, October 2008, <https://www.rfc-editor.org/info/rfc5309>.

[RFC5311] McPherson, D., Ed., Ginsberg, L., Previdi, S., and M. Shand, "Simplified Extension of Link State PDU (LSP) Space for IS-IS", RFC 5311, DOI 10.17487/RFC5311, February 2009, <https://www.rfc-editor.org/info/rfc5311>.

[RFC5316] Chen, M., Zhang, R., and X. Duan, "ISIS Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5316, DOI 10.17487/RFC5316, December 2008, <https://www.rfc-editor.org/info/rfc5316>.

[RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <https://www.rfc-editor.org/info/rfc7356>.

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <https://www.rfc-editor.org/info/rfc7981>.

9.2. Informative References

[I-D.ietf-isis-segment-routing-extensions] Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-16 (work in progress), April 2018.

[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.

[RFC3277] McPherson, D., "Intermediate System to Intermediate System (IS-IS) Transient Blackhole Avoidance", RFC 3277, DOI 10.17487/RFC3277, April 2002, <https://www.rfc-editor.org/info/rfc3277>.

[RFC3719] Parker, J., Ed., "Recommendations for Interoperable Networks using Intermediate System to Intermediate System (IS-IS)", RFC 3719, DOI 10.17487/RFC3719, February 2004, <https://www.rfc-editor.org/info/rfc3719>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[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>.

[RFC5449] Baccelli, E., Jacquet, P., Nguyen, D., and T. Clausen, "OSPF Multipoint Relay (MPR) Extension for Ad Hoc Networks", RFC 5449, DOI 10.17487/RFC5449, February 2009, <https://www.rfc-editor.org/info/rfc5449>.

[RFC5614] Ogier, R. and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding", RFC 5614, DOI 10.17487/RFC5614, August 2009, <https://www.rfc-editor.org/info/rfc5614>.

[RFC5837] Atlas, A., Ed., Bonica, R., Ed., Pignataro, C., Ed., Shen, N., and JR. Rivers, "Extending ICMP for Interface and Next-Hop Identification", RFC 5837, DOI 10.17487/RFC5837, April 2010, <https://www.rfc-editor.org/info/rfc5837>.

[RFC6232] Wei, F., Qin, Y., Li, Z., Li, T., and J. Dong, "Purge Originator Identification TLV for IS-IS", RFC 6232, DOI 10.17487/RFC6232, May 2011, <https://www.rfc-editor.org/info/rfc6232>.

[RFC7182] Herberg, U., Clausen, T., and C. Dearlove, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)", RFC 7182, DOI 10.17487/RFC7182, April 2014, <https://www.rfc-editor.org/info/rfc7182>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/info/rfc7921>.

Authors' Addresses

Russ White (editor)
LinkedIn

Email: russ@riw.us

Shawn Zandi (editor)
LinkedIn

Email: szandi@linkedin.com