

Network Working Group
Internet-Draft
Obsoletes: 4970 (if approved)
Intended status: Standards Track
Expires: February 1, 2015

A. Lindem, Ed.
N. Shen
J. Vasseur
Cisco Systems
R. Aggarwal
Arktan
S. Shaffer
Akamai
July 31, 2014

Extensions to OSPF for Advertising Optional Router Capabilities
draft-acee-ospf-rfc4970bis-00.txt

Abstract

It is useful for routers in an OSPFv2 or OSPFv3 routing domain to know the capabilities of their neighbors and other routers in the routing domain. This document proposes extensions to OSPFv2 and OSPFv3 for advertising optional router capabilities. A new Router Information (RI) Link State Advertisement (LSA) is proposed for this purpose. In OSPFv2, the RI LSA will be implemented with a new opaque LSA type ID. In OSPFv3, the RI LSA will be implemented with a new LSA type function code. In both protocols, the RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). This document obsoletes RFC 4970 by providing a revised specification including support for advertisement of multiple instances of the RI LSA and a TLV for functional capabilities.

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 February 1, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Requirements Notation	3
1.2. Summary of Changes from RFC 4970	3
2. OSPF Router Information (RI) LSA	4
2.1. OSPFv2 Router Information (RI) Opaque LSA	4
2.2. OSPFv3 Router Information (RI) Opaque LSA	6
2.3. OSPF Router Information Capabilities TLV	6
2.4. Assigned OSPF Router Information Capability Bits	8
2.5. OSPF Router Functional Capabilities TLV	8
2.6. Flooding Scope of the Router Information LSA	9
3. Security Considerations	11
4. IANA Considerations	12
5. References	15
5.1. Normative References	15
5.2. Informative References	15
Appendix A. Acknowledgments	16
Authors' Addresses	17

1. Introduction

It is useful for routers in an OSPFv2 [OSPF] or OSPFv3 [OSPFV3] routing domain to know the capabilities of their neighbors and other routers in the routing domain. This can be useful for both the advertisement and discovery of OSPFv2 and OSPFv3 capabilities. Throughout this document, OSPF will be used when the specification is applicable to both OSPFv2 and OSPFv3. Similarly, OSPFv2 or OSPFv3 will be used when the text is protocol specific.

OSPF uses the options field in LSAs and hello packets to advertise optional router capabilities. In the case of OSPFv2, all the bits in this field have been allocated so new optional capabilities cannot be advertised. This document proposes extensions to OSPF to advertise these optional capabilities via opaque LSAs in OSPFv2 and new LSAs in OSPFv3. For existing OSPF capabilities, backward-compatibility issues dictate that this advertisement is used primarily for informational purposes. For future OSPF extensions, this advertisement MAY be used as the sole mechanism for advertisement and discovery.

This document obsoletes RFC 4970 by providing a revised specification including support for advertisement of multiple instances of the RI LSA and a TLV for functional capabilities.

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 [RFC-KEYWORDS].

1.2. Summary of Changes from RFC 4970

This document includes the following changes from RFC 4970 [RFC4970]:

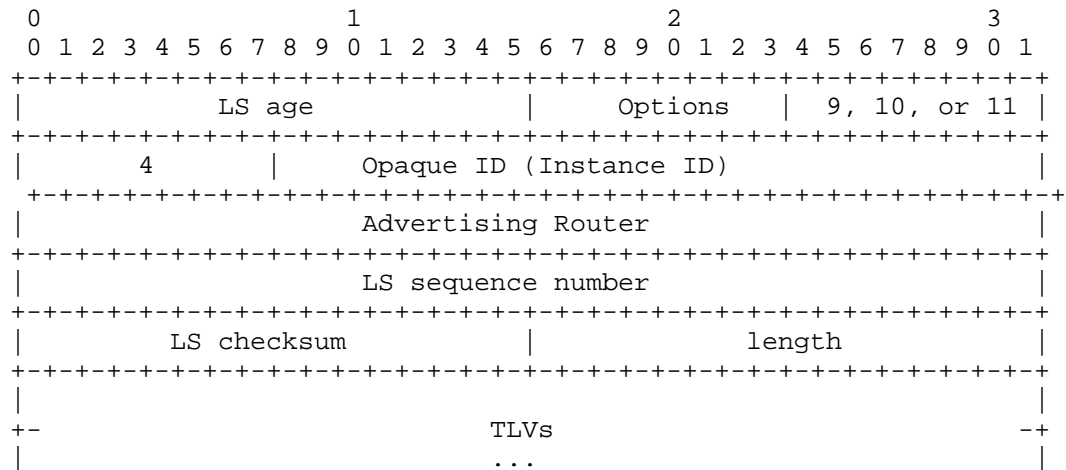
1. The main change is that an OSPF router will be able to advertise multiple instances of the OSPF Router Information LSA. This change permeates through much of the document
2. Additionally, Section 2.5 includes a new TLV for functional capabilities. This is constast to the existing TLV which is used to advertise capabilities for informational purposes only.
3. Finally, references have been updated for drafts that have become RFCs and RFCs that have been obseleted since the publication of RFC 4970.

2. OSPF Router Information (RI) LSA

OSPFv2 routers will advertise a link scoped, area-scoped, or AS-scoped Opaque-LSA [OPAQUE]. The OSPFv2 Router Information LSA has an Opaque type of 4 and Opaque ID is the instance ID. The first instance ID, i.e., 0, should always contain the Router Information Capabilities TLV and, if advertised, the Router Functional Capabilities TLV. RI Information LSAs subsequent to the first can be used for information which doesn't fit in the first instance.

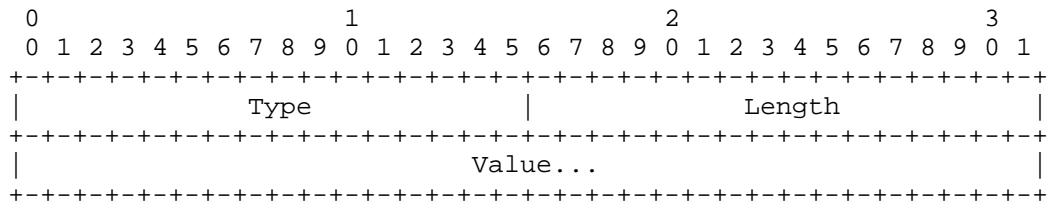
2.1. OSPFv2 Router Information (RI) Opaque LSA

OSPFv2 routers will advertise a link scoped, area-scoped, or AS-scoped Opaque-LSA [OPAQUE]. The OSPFv2 Router Information LSA has an Opaque type of 4 and Opaque ID specifies the LSA instance ID with the first instance always having an Instance ID of 0.



OSPFv2 Router Information Opaque LSA

The format of the TLVs within the body of an RI LSA is the same as the format used by the Traffic Engineering Extensions to OSPF [TE]. The LSA payload consists of one or more nested Type/Length/Value (TLV) triplets. The format of each TLV is:

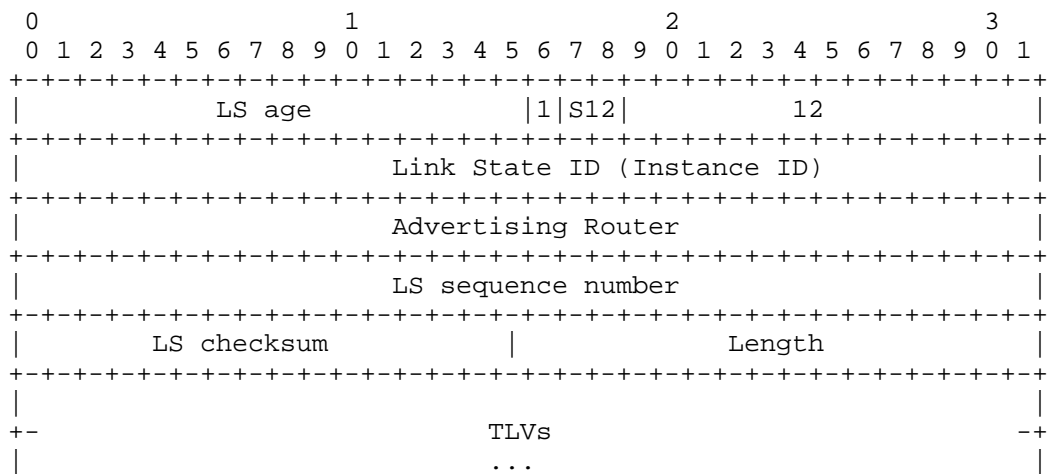


TLV Format

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-byte 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. Unrecognized types are ignored.

2.2. OSPFv3 Router Information (RI) Opaque LSA

The OSPFv3 Router Information LSA has a function code of 12 while the S1/S2 bits are dependent on the desired flooding scope for the LSA. The U bit will be set indicating that the OSPFv3 RI LSA should be flooded even if it is not understood. The Link State ID (LSID) value for this LSA is the instance ID. The first instance ID, i.e., 0, should always contain the Router Information Capabilities TLV and, if advertised, the Router Functional Capabilities TLV. OSPFv3 Router Information LSAs subsequent to the first can be used for information which doesn't fit in the first instance. OSPFv3 routers MAY advertise multiple RIs LSA per flooding scope.



OSPFv3 Router Information LSA

The format of the TLVs within the body of an RI LSA is as defined in Section 2.1

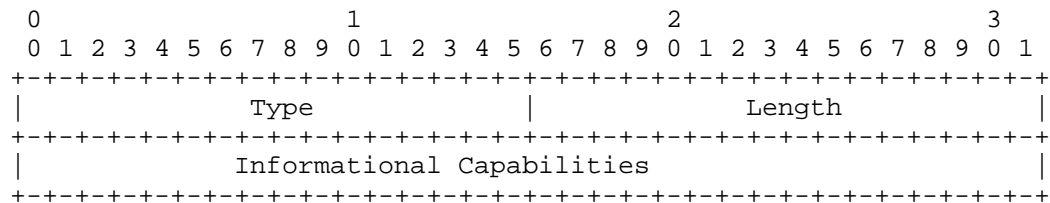
When a new Router Information LSA TLV is defined, the specification MUST explicitly state whether the TLV is applicable to OSPFv2 only, OSPFv3 only, or both OSPFv2 and OSPFv3.

2.3. OSPF Router Information Capabilities TLV

The first defined TLV in the body of an RI LSA is the Router Information Capabilities TLV. An OSPF router advertising an OSPF RI LSA MAY include the Router Information Capabilities TLV. If included, it MUST be the first TLV in the first instance of the OSPF RI LSA. Additionally, the TLV MUST accurately reflect the OSPF

router's capabilities in the scope advertised. However, the informational capabilities advertised have no impact on the OSPF's operation -- they are advertised purely for informational purposes.

The format of the Router Informational Capabilities TLV is as follows:



Type A 16-bit field set to 1.

Length A 16-bit field that indicates the length of the value portion in octets and will be a multiple of 4 octets dependent on the number of capabilities advertised. Initially, the length will be 4, denoting 4 octets of informational capability bits.

Value A variable length sequence of capability bits rounded to a multiple of 4 octets padded with undefined bits. Initially, there are 4 octets of capability bits. Bits are numbered left-to-right starting with the most significant bit being bit 0.

OSPF Router Informational Capabilities TLV

The Router Informational Capabilities TLV MAY be followed by optional TLVs that further specify a capability.

2.4. Assigned OSPF Router Informational Capability Bits

The following informational capability bits are assigned:

Bit	Capabilities
0	OSPF graceful restart capable [GRACE]
1	OSPF graceful restart helper [GRACE]
2	OSPF Stub Router support [STUB]
3	OSPF Traffic Engineering support [TE]
4	OSPF point-to-point over LAN [P2PLAN]
5	OSPF Experimental TE [EXP-TE]
6-31	Unassigned (Standards Action)

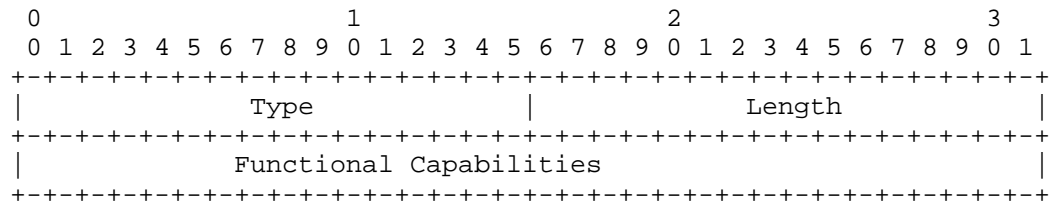
OSPF Router Informational Capabilities Bits

References for [GRACE], [STUB], [TE], [P2PLAN], and [EXP-TE] are included herein.

2.5. OSPF Router Functional Capabilities TLV

This specification also defines the Router Functional Capabilities TLV for advertisement within the OSPF Router Information LSA. An OSPF router advertising an OSPF RI LSA MAY include the Router Functional Capabilities TLV. If included, it MUST be the included in the first instance of the LSA. Additionally, the TLV MUST be used to reflect OSPF router functional capabilities. If the TLV is not included or the length doesn't include the assigned OSPF functional capability bit, the corresponding OSPF functional capability is implicitly advertised as not being support by the advertising OSPF router.

The format of the Router Functional Capabilities TLV is as follows:



Type A 16-bit field set to 1.

Length A 16-bit field that indicates the length of the value portion in octets and will be a multiple of 4 octets dependent on the number of capabilities advertised. Initially, the length will be 4, denoting 4 octets of informational capability bits.

Value A variable length sequence of capability bits rounded to a multiple of 4 octets padded with undefined bits. Initially, there are 4 octets of capability bits. Bits are numbered left-to-right starting with the most significant bit being bit 0.

OSPF Router Functional Capabilities TLV

The Router Functional Capabilities TLV MAY be followed by optional TLVs that further specify a capability. In contrast to the Router Information Capabilities TLV, the OSPF extensions advertised in this TLV MAY be used to by other OSPF routers to dictate protocol operation. The specifications for functional capabilities advertised in this TLV MUST describe protocol behavior and address backward compatibility.

2.6. Flooding Scope of the Router Information LSA

The flooding scope for a Router Information LSA is determined by the LSA type. For OSPFv2, type 9 (link-scoped), type 10 (area-scoped), or a type 11 (AS-scoped) opaque LSA may be flooded. For OSPFv3, the S1 and S2 bits in the LSA type determine the flooding scope. If AS-wide flooding scope is chosen, the originating router should also advertise area-scoped LSA(s) into any attached Not-So-Stubby Area (NSSA) area(s). An OSPF router MAY advertise different capabilities when both NSSA area scoped LSA(s) and an AS-scoped LSA are advertised. This allows functional capabilities to be limited in scope. For example, a router may be an area border router but only support traffic engineering (TE) in a subset of its attached areas.

The choice of flooding scope is made by the advertising router and is

a matter of local policy. The originating router MAY advertise multiple RI LSAs as long as the flooding scopes differ. TLV flooding scope rules will be specified on a per-TLV basis and MUST be specified in the accompanying specifications for new Router Information LSA TLVs.

3. Security Considerations

This document describes both a generic mechanism for advertising router capabilities and a TLV for advertising informational and functional capability bits. The capability TLVs are less critical than the topology information currently advertised by the base OSPF protocol. The security considerations for the generic mechanism are dependent on the future application and, as such, should be described as additional capabilities are proposed for advertisement. Security considerations for the base OSPF protocol are covered in [OSPF] and [OSPFV3].

4. IANA Considerations

The following IANA assignment was made from an existing registry:

The OSPFv2 opaque LSA type 4 has been reserved for the OSPFv2 RI opaque LSA.

The following registries have been defined for the following purposes:

1. Registry for OSPFv3 LSA Function Codes - This new top-level registry will be comprised of the fields Value, LSA function code name, and Document Reference. The OSPFv3 LSA function code is defined in section A.4.2.1 of [OSPFV3]. The OSPFv3 LSA function code 12 has been reserved for the OSPFv3 Router Information (RI) LSA.

Range	Assignment Policy
0	Reserved (not to be assigned)
1-9	Already assigned
10-11	Unassigned (Standards Action)
12	OSPFv3 RI LSA (Assigned herein)
13-255	Unassigned (Standards Action)
256-8175	Reserved (No assignments)
8176-8183	Experimentation (No assignments)
8184-8191	Vendor Private Use (No assignments)

OSPFv3 LSA Function Codes

- * OSPFv3 LSA function codes in the range 256-8175 are not to be assigned at this time. Before any assignments can be made in this range, there MUST be a Standards Track RFC that specifies IANA Considerations that cover the range being assigned.
- * OSPFv3 LSA function codes in the range 8176-8181 are for experimental use; these will not be registered with IANA and MUST NOT be mentioned by RFCs.

- * OSPFv3 LSAs with an LSA Function Code in the Vendor Private Use range 8184-8191 MUST include the Vendor Enterprise Code as the first 4 octets following the 20 octets of LSA header.
 - * If a new LSA Function Code is documented, the documentation MUST include the valid combinations of the U, S2, and S1 bits for the LSA. It SHOULD also describe how the Link State ID is to be assigned.
2. Registry for OSPF RI TLVs - This top-level registry will be comprised of the fields Value, TLV Name, and Document Reference. The value of 1 for the capabilities TLV is defined herein.

Range	Assignment Policy
0	Reserved (not to be assigned)
1	Already assigned
2-32767	Unassigned (Standards Action)
32768-32777	Experimentation (No assignments)
32778-65535	Reserved (Not to be assigned)

OSPF RI TLVs

- * Types in the range 32768-32777 are for experimental use; these will not be registered with IANA and MUST NOT be mentioned by RFCs.
 - * Types in the range 32778-65535 are reserved and are not to be assigned at this time. Before any assignments can be made in this range, there MUST be a Standards Track RFC that specifies IANA Considerations that covers the range being assigned.
3. Registry for OSPF Router Informational Capability Bits - This sub-registry of the OSPF RI TLV registry will be comprised of the fields Bit Number, Capability Name, and Document Reference. The values are defined in Section 2.4. All Router Informational Capability TLV additions are to be assigned through standards action.
4. Registry for OSPF Router Functional Capability Bits - This sub-registry of the OSPF RI TLV registry will be comprised of the fields Bit Number, Capability Name, and Document Reference.

Initially, the sub-registry will be empty but will be available for future capabilities. All Router Functional Capability TLV additions are to be assigned through standards action.

5. References

5.1. Normative References

- [OPAQUE] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.
- [OSPF] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [OSPFV3] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.
- [RFC-KEYWORDS] Bradner, S., "Key words for use in RFC's to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4970] Lindem, A., Shen, N., Vasseur, J., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.
- [TE] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering Extensions to OSPF", RFC 3630, September 2003.

5.2. Informative References

- [EXP-TE] Srisuresh, P. and P. Joseph, "OSPF-xTE: Experimental Extension to OSPF for Traffic Engineering", RFC 4973, July 2007.
- [GRACE] Moy, J., Pillay-Esnault, P., and A. Lindem, "Graceful OSPF Restart", RFC 3623, November 2003.
- [P2PLAN] Shen, N. and A. Zinin, "Point-to-point operation over LAN in link-state routing protocols", RFC 5309, October 2008.
- [STUB] Retana, A., Nguyen, L., White, R., Zinin, A., and D. McPherson, "OSPF Stub Router Advertisement", RFC 6987, September 2013.

Appendix A. Acknowledgments

The idea for this work grew out of a conversation with Andrew Partan and we would like to thank him for his contribution. The authors would like to thanks Peter Psenak for his review and helpful comments on early versions of the document.

Comments from Abhay Roy, Vishwas Manral, Vivek Dubey, and Adrian Farrel have been incorporated into later versions.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Naiming Shen
Cisco Systems
225 West Tasman Drive
San Jose, CA 95134
USA

Email: naiming@cisco.com

Jean-Philippe Vasseur
Cisco Systems
1414 Massachusetts Avenue
Boxborough, MA 01719
USA

Email: jpv@cisco.com

Rahul Aggarwal
Arktan

Email: raggarwa_1@yahoo.com

Scott Shaffer
Akamai
8 Cambridge Center
Cambridge, MA 02142
USA

Email: sshaffer@akamai.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2015

H. Chen
R. Li
A. Kumar S N
Huawei Technologies
G. Cauchie

A. Retana
Cisco Systems, Inc.
N. So
Tata Communications
V. Liu
China Mobile
M. Toy
Comcast
L. Liu
UC Davis
October 26, 2014

OSPF Topology-Transparent Zone
draft-chen-ospf-ttz-09.txt

Abstract

This document presents a topology-transparent zone in a domain. A topology-transparent zone comprises a group of routers and a number of links connecting these routers. Any router outside of the zone is not aware of the zone. The information about the links and routers inside the zone is not distributed to any router outside of the zone. Any link state change such as a link down inside the zone is not seen by any router outside of the zone.

Status of this Memo

This Internet-Draft is submitted to IETF 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 April 29, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Conventions Used in This Document	5
3. Requirements	5
4. Topology-Transparent Zone	5
4.1. Overview of Topology-Transparent Zone	5
4.2. An Example of TTZ	6
5. Extensions to OSPF Protocols	7
5.1. Opaque LSAs for TTZ	7
5.2. A TTZ Capability TLV in Router Information LSA	10
6. Constructing LSAs for TTZ	11
7. Establishing Adjacencies	12
7.1. Discover TTZ Neighbor over Normal Adjacency	12
7.2. Establishing TTZ Adjacencies	12
7.3. Adjacency between TTZ Edge and Router outside	13
8. Distribution of LSAs	13
8.1. Distribution of LSAs within TTZ	14
8.2. Distribution of LSAs through TTZ	14
9. Computation of Routing Table	14
10. Operations	14
10.1. Configuring TTZ	14
10.2. Smooth Migration to TTZ	15
10.3. Adding a Router into TTZ	16
11. Prototype Implementation	16
11.1. What are Implemented and Tested	16
11.2. Implementation Experience	18
12. Security Considerations	18
13. IANA Considerations	19
14. Contributors	19
15. Acknowledgement	19
16. References	19
16.1. Normative References	19
16.2. Informative References	19
Authors' Addresses	20

1. Introduction

The number of routers in a network becomes larger and larger as the Internet traffic keeps growing. Through splitting the network into multiple areas, we can extend the network further. However, there are a number of issues when a network is split further into more areas.

At first, dividing a network from one area into multiple areas or from a number of existing areas to even more areas is a very challenging and time consuming task since it is involved in significant network architecture changes. Considering the one area case, originally the network has only one area, which is the backbone. This original backbone area will be split into a new backbone and a number of non backbone areas. In general, each of the non backbone areas is connected to the new backbone area through the area border routers between the non backbone and the backbone area. There is not any direct connection between any two non backbone areas. Each area border router summarizes the topology of its attached non backbone area for transmission on the backbone area, and hence to all other area border routers.

Secondly, the services carried by the network may be interrupted while the network is being split from one area into multiple areas or from a number of existing areas into even more areas.

Furthermore, it is complex for a Multi-Protocol Label Switching (MPLS) Traffic Engineering (TE) Label Switching Path (LSP) crossing multiple areas to be setup. In one option, a TE path crossing multiple areas is computed by using collaborating Path Computation Elements (PCEs) [RFC5441] through the PCE Communication Protocol (PCEP)[RFC5440], which is not easy to configure by operators since the manual configuration of the sequence of domains is required. Although this issue can be addressed by using the Hierarchical PCE, this solution may further increase the complexity of network design. Especially, the current PCE standard method may not guarantee that the path found is optimal.

This document presents a topology-transparent zone in an area and describes extensions to OSPF for supporting the topology-transparent zone, which is scalable and resolves the issues above.

A topology-transparent zone comprises a group of routers and a number of links connecting these routers. Any router outside of the zone is not aware of the zone. The information about the links and routers inside the zone is not distributed to any router outside of the zone. Any link state change such as a link down inside the zone is not seen by any router outside of the zone.

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 RFC 2119.

3. Requirements

Topology-Transparent Zone (TTZ) may be deployed for resolving some critical issues in existing networks and future networks. The requirements for TTZ are listed as follows:

- o TTZ MUST be backward compatible. When a TTZ is deployed on a set of routers in a network, the routers outside of the TTZ in the network do not need to know or support TTZ.
- o TTZ MUST support at least one more levels of network hierarchies, in addition to the hierarchies supported by existing routing protocols.
- o Users SHOULD be able to easily set up an end to end service crossing TTZs.
- o The configuration for a TTZ in a network SHOULD be minimum.
- o The changes on the existing protocols for supporting TTZ SHOULD be minimum.

4. Topology-Transparent Zone

4.1. Overview of Topology-Transparent Zone

A Topology-Transparent Zone (TTZ) is identified by an Identifier (ID), and it includes a group of routers and a number of links connecting the routers. A TTZ is in an OSPF area.

The ID of a TTZ or TTZ ID is a number that is unique for identifying an entity such as a node in an OSPF domain. It is not zero in general.

In addition to having the functions of an OSPF area, an OSPF TTZ makes some improvements on an OSPF area, which include:

- o An OSPF TTZ is virtualized as the TTZ edge routers connected.

- o An OSPF TTZ receives the link state information about the topology outside of the TTZ, stores the information in the TTZ and floods the information through the TTZ to the routers outside of the TTZ.

4.2. An Example of TTZ

The figure below shows an area containing a TTZ: TTZ 600.

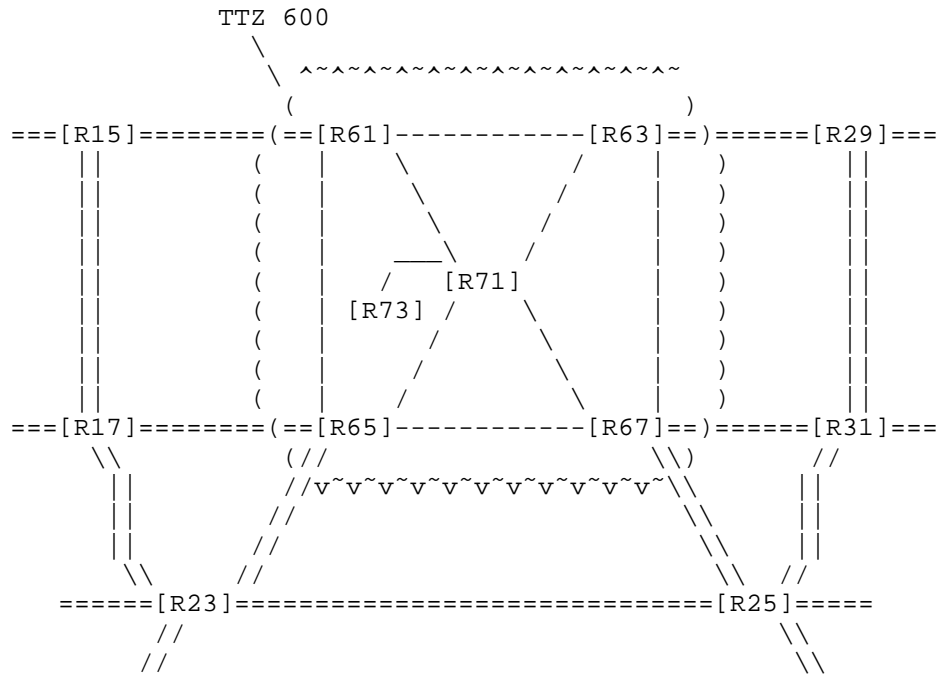


Figure 1: An Example of TTZ

The area comprises routers R15, R17, R23, R25, R29 and R31. It also contains TTZ 600, which comprises routers R61, R63, R65, R67, R71 and R73, and the links connecting them.

There are two types of routers in a TTZ: TTZ internal routers and TTZ edge routers. A TTZ internal router is a router inside the TTZ and its adjacent routers are in the TTZ. A TTZ edge router is a router inside the TTZ and has at least one adjacent router that is outside of the TTZ.

The TTZ in the figure above comprises four TTZ edge routers R61, R63, R65 and R67. Each TTZ edge router is connected to at least one router outside of the TTZ. For instance, router R61 is a TTZ edge

router since it is connected to router R15, which is outside of the TTZ.

In addition, the TTZ comprises two TTZ internal routers R71 and R73. A TTZ internal router is not connected to any router outside of the TTZ. For instance, router R71 is a TTZ internal router since it is not connected to any router outside of the TTZ. It is just connected to routers R61, R63, R65, R67 and R73 in the TTZ.

A TTZ MUST hide the information inside the TTZ from the outside. It MUST NOT directly distribute any internal information about the TTZ to a router outside of the TTZ.

For instance, the TTZ in the figure above MUST NOT send the information about TTZ internal router R71 to any router outside of the TTZ in the routing domain; it MUST NOT send the information about the link between TTZ router R61 and R65 to any router outside of the TTZ.

In order to create a TTZ, we MUST configure the same TTZ ID on the edge routers and identify the TTZ internal links on them. In addition, we SHOULD configure the TTZ ID on every TTZ internal router which indicates that every link of the router is a TTZ internal link.

From a router outside of the TTZ, a TTZ is seen as a group of routers fully connected. For instance, router R15 in the figure above, which is outside of TTZ 600, sees TTZ 600 as a group of TTZ edge routers: R61, R63, R65 and R67. These four TTZ edge routers are fully connected.

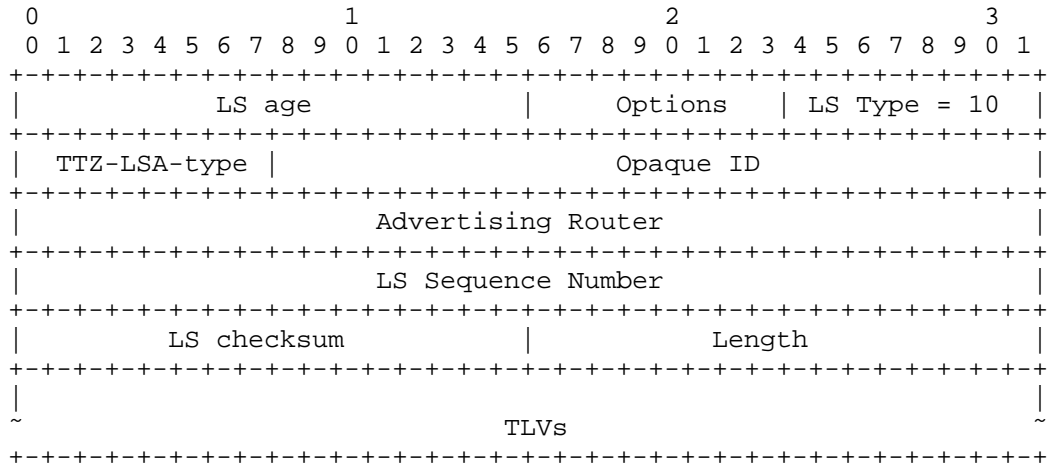
In addition, a router outside of the TTZ sees TTZ edge routers having normal connections to the routers outside of the TTZ. For example, router R15 sees four TTZ edge routers R61, R63, R65 and R67, which have the normal connections to R15, R29, R17 and R23, R25 and R31 respectively.

5. Extensions to OSPF Protocols

5.1. Opaque LSAs for TTZ

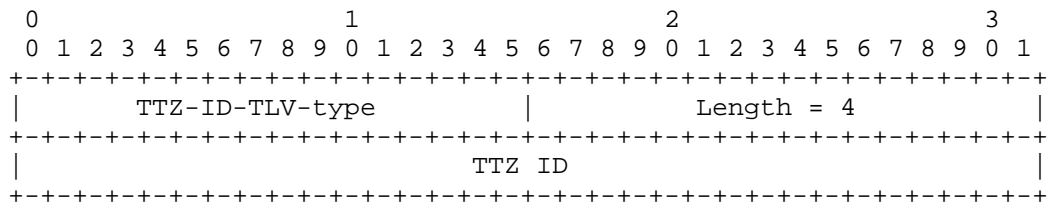
The link state information about a TTZ includes router LSAs and network LSAs describing the TTZ topology. These LSAs can be contained and distributed in opaque LSAs within the TTZ. Some control information on a TTZ can also be contained and distributed in opaque LSAs within the TTZ. These opaque LSAs are called TTZ opaque LSAs or TTZ LSAs for short.

The following is a general form of a TTZ LSA. It has an LS type = 10 and TTZ-LSA-Type, and contains a number of TLVs.

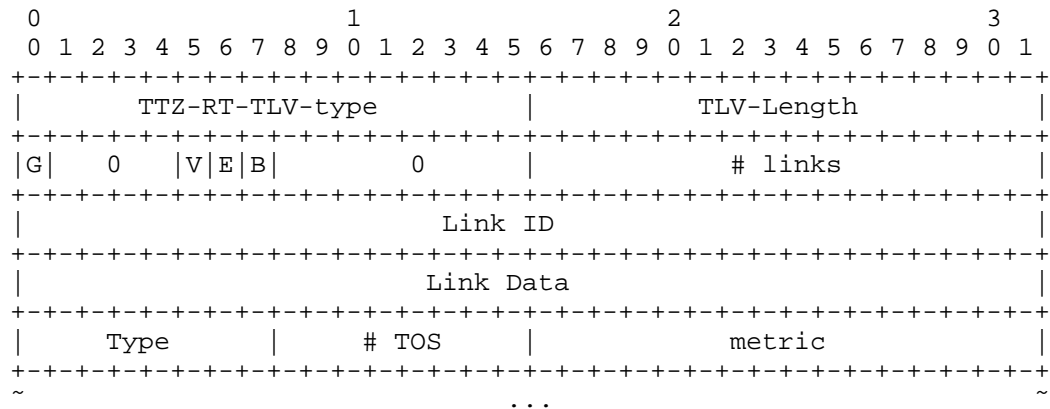


Where TTZ-LSA-type may be TBD1 (TTZ-RT-LSA-type) for TTZ Router LSA, TBD2 (TTZ-NW-LSA-type) for TTZ Network LSA, and TBD3 (TTZ-CT-LSA-type) for TTZ Control LSA.

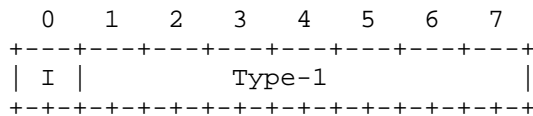
There are four types of TLVs: TTZ ID TLV, TTZ Router TLV, TTZ network TLV and TTZ Options TLV. A TTZ ID TLV has the following format. It contains a TTZ ID.



The format of a TTZ Router TLV is as follows. It contains the contents of a normal router LSA. A TTZ router LSA includes a TTZ ID TLV and a TTZ Router TLV.



Where G = 1/0 indicates that the router is an edge/internal router of TTZ. For a router link, the existing eight bit Type field for a router link may be split into two fields as follows:



I bit flag:

1: Router link is an internal link to a router inside TTZ.

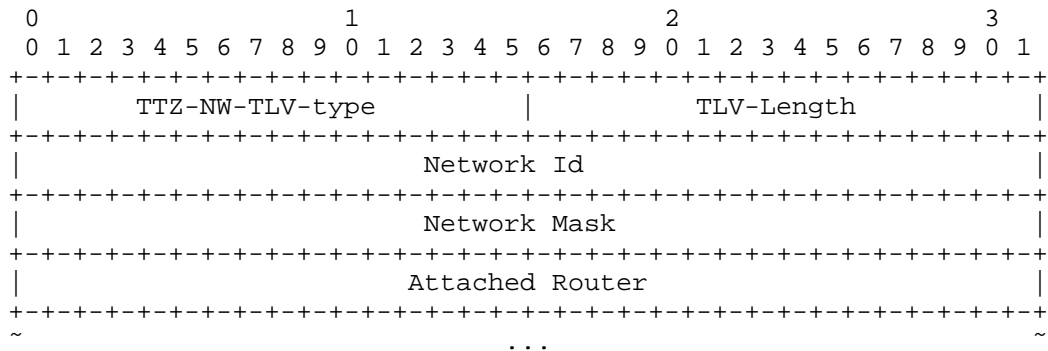
0: This indicates that the router link is an external link.

Type-1: The kind of the link.

For a link inside a TTZ, I bit flag is set to one, indicating that this link is an internal TTZ link. For a link connecting to a router outside of a TTZ from a TTZ edge router, I bit flag is set to zero, indicating that this link is an external TTZ link.

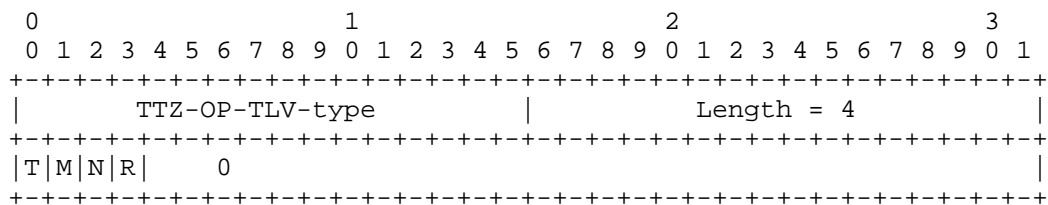
The value of Type-1 may be 1, 2, 3, or 4, which indicates that the kind of a link being described is a point-to-point connection to another router, a connection to a transit network, a connection to a stub network, or a virtual link respectively.

A TTZ Network TLV has the following format. It contains the contents of a normal network LSA. A TTZ network LSA includes a TTZ ID TLV and a TTZ network TLV.



Where Network ID is the interface address of the DR, which is followed by the contents of a network LSA.

The format of TTZ Options TLV is as follows. A TTZ control LSA contains a TTZ ID TLV and a TTZ Options TLV.



T = 1: Distributing TTZ Topology Information for Migration

M = 1: Migrating to TTZ

N = 1: Distributing Normal Topology Information for Rollback

R = 1: Rolling back from TTZ

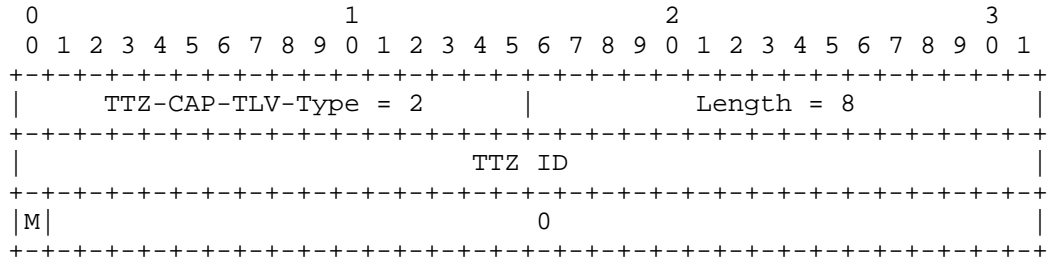
5.2. A TTZ Capability TLV in Router Information LSA

A new bit such as bit 6 for TTZ capability may be defined in the Router Informational Capabilities TLV as follows:

Bit	Capabilities
0	OSPF graceful restart capable [GRACE]
:	...
5	OSPF Experimental TE [EXP-TE]
6	OSPF TTZ capable [OSPF-TTZ]
7-31	Unassigned (Standards Action)

When the OSPF TTZ capable bit is set to one, a TTZ capability TLV must follow the Router Informational Capabilities TLV to indicate a link/router's TTZ capability and the TTZ to which the link/router

belongs. It has the following format.



It contains a TTZ ID and a number of TTZ bits. The following bits in the TLV are assigned:

Bit	Meaning
M	Have Migrated to TTZ (i.e., works as TTZ)
1-31	Unassigned (Standards Action)

A link scope RI LSA with a OSPF TTZ capable bit set to one and a TTZ Capability TLV will be used to discover a TTZ neighbor.

6. Constructing LSAs for TTZ

There are three types of LSAs for representing a TTZ: TTZ router LSA, TTZ network LSA and Router LSA for virtualizing TTZ. The first two may be generated by a TTZ router, and the third by a TTZ edge router.

A TTZ router LSA generated by a TTZ router has a TTZ ID TLV and a TTZ Router TLV. The former includes the ID of the TTZ to which the router belongs. The latter contains the links to the router.

A TTZ network LSA for a broadcast link is generated by the DR for the link. It contains a TTZ ID TLV and a TTZ network TLV. The former has the ID of the TTZ to which the link belongs. The latter includes the DR's address, the network mask, and the routers attached.

A router LSA for virtualizing a TTZ generated by an edge router of the TTZ comprises three groups of links in general.

The first group are the router links connecting the routers outside of the TTZ. These router links are normal router links. There is a router link for every adjacency between this TTZ edge router and a router outside of the TTZ.

The second group are the "virtual" router links. For each of the other TTZ edge routers, there is a point-to-point router link to it.

The cost of the link may be the cost of the shortest path from this TTZ edge router to it within the TTZ.

In addition, the LSA may contain a third group of links, which are stub links for other destinations inside the TTZ. They may be the loopback addresses to be accessed by a node outside of the TTZ.

7. Establishing Adjacencies

This section describes the adjacencies in some different cases.

7.1. Discover TTZ Neighbor over Normal Adjacency

For two routers A and B connected by a P2P link and having a normal adjacency, they discover TTZ each other through a link scope RI LSA with an OSPF TTZ capable bit and a TTZ ID. We call this LSA D-LSA for short. If two ends of the link have the same TTZ ID, A and B are TTZ neighbors. The following is a sequence of events related to TTZ.

```

      A                                     B
  Configure TTZ                           Configure TTZ
      D-LSA (TTZ-ID=100)
      -----> Same TTZ ID
                                     A is B's TTZ Neighbor
      D-LSA (TTZ-ID=100)
  Same TTZ ID <-----
  B is A's TTZ Neighbor

```

A sends B a D-LSA with TTZ-ID after the TTZ is configured on it. B sends A a D-LSA with TTZ-ID after the TTZ is configured on it. When A receives the D-LSA from B and determines they have the same TTZ ID, B is A's TTZ neighbor. When B receives the D-LSA from A and determines they have the same TTZ ID, A is B's TTZ neighbor.

For a number of routers connected through a broadcast link and having normal adjacencies among them, they also discover TTZ each other through D-LSAs. The DR for the link "forms" TTZ adjacency with each of the other routers if all the routers attached to the link have the same TTZ ID configured on the connections to the link.

7.2. Establishing TTZ Adjacencies

When a router (say A) is connected via a P2P link to another router (say B) and there is not any adjacency between them over the link, a user configures TTZ on two ends of the link to form a TTZ adjacency.

While A and B are forming an adjacency, they start to discover TTZ each other through D-LSAs in the same way as described above after the normal adjacency is greater than ExStart. When the normal adjacency is full and B becomes A's TTZ neighbor, A forms a TTZ adjacency with B. Similarly, B forms a TTZ adjacency with A.

For a number of routers connected through a broadcast link and having no adjacency among them, they start to form TTZ adjacencies after TTZ is configured on the link. While forming adjacencies, they discover TTZ each other through D-LSAs in the same way as described above after the normal adjacency is greater than ExStart. The DR for the link forms TTZ adjacency with each of the other routers if all the routers attached to the link have the same TTZ ID configured on the connections to the link. Otherwise, the DR does not form any adjacency with any router attached to the link.

An alternative way for forming an adjacency between two routers in a TTZ is to extend hello protocol. Hello protocol is extended to include TTZ ID in LLS of a hello packet. The procedure for handling hellos is changed to consider TTZ ID. If two routers have the same TTZ IDs in their hellos, an adjacency between these two routers is to be formed; otherwise, no adjacency is formed.

7.3. Adjacency between TTZ Edge and Router outside

For an edge router in a TTZ, it forms an adjacency with any router outside of the TTZ that has a connection with it.

When the edge router synchronizes its link state database with the router outside of the TTZ, it sends the router outside of the TTZ the information about all the LSAs except for the LSAs belonging to the TTZ that are hidden from any router outside of the TTZ.

At the end of the link state database synchronization, the edge router originates its own router LSA for virtualizing the TTZ and sends this LSA to the router outside of the TTZ.

From the point of view of the router outside of the TTZ, it sees the other end as a normal router and forms the adjacency in the same way as a normal router. It is not aware of anything about its neighboring TTZ. From the LSAs related to the TTZ edge router in the other end, it knows that the TTZ edge router is connected to each of the other TTZ edge routers and some routers outside of the TTZ.

8. Distribution of LSAs

LSAs can be divided into a couple of classes according to their

distributions. The first class of LSAs is distributed within a TTZ. The second is distributed through a TTZ.

8.1. Distribution of LSAs within TTZ

Any LSA about a link state in a TTZ is distributed within the TTZ. It is not distributed to any router outside of the TTZ. For example, a router LSA generated for a router in a TTZ is distributed within the TTZ and not distributed to any router outside of the TTZ.

Any network LSA generated for a broadcast or NBMA network in a TTZ is distributed in the TTZ and not sent to a router outside of the TTZ.

Any opaque LSA generated for a TTZ internal TE link is distributed within the TTZ and not distributed to any router outside of the TTZ.

8.2. Distribution of LSAs through TTZ

Any LSA about a link state outside of a TTZ received by an edge router of the TTZ is distributed through the TTZ. For example, when an edge router of a TTZ receives an LSA from a router outside of the TTZ, it floods it to its neighboring routers both inside the TTZ and outside of the TTZ. This LSA may be any LSA such as a router LSA that is distributed in a domain.

The routers in the TTZ continue to flood the LSA. When another edge router of the TTZ receives the LSA, it floods the LSA to its neighboring routers both outside of the TTZ and inside the TTZ.

9. Computation of Routing Table

The computation of the routing table on a router is the same as that described in RFC 2328, with one exception. A router in a TTZ MUST ignore the router LSAs generated by the edge routers of the TTZ for virtualizing the TTZ. It computes routes through using the TTZ topology represented by TTZ LSAs and the topology outside of the TTZ.

10. Operations

10.1. Configuring TTZ

This section proposes some options for configuring a TTZ.

1. Configuring TTZ on Every Link in TTZ

If every link in a TTZ is configured with a same TTZ ID as a TTZ

link, the TTZ is determined. A router with some TTZ links and some normal links is a TTZ edge router. A router with only TTZ links is a TTZ internal router.

2. Configuring TTZ on Every Router in TTZ

We may configure a same TTZ ID on every router in the TTZ, and on every edge router's links connecting to the routers in the TTZ.

A router configured with the TTZ ID on some of its links is a TTZ edge router. A router configured with the TTZ ID only is a TTZ internal router. All the links on a TTZ internal router are TTZ links. This option is simpler than the above one.

10.2. Smooth Migration to TTZ

For a group of routers and a number of links connecting the routers in an area, making them transfer to work as a TTZ without any service interruption may take a few of steps or stages.

At first, users configure the TTZ feature on every router in the TTZ. In this stage, a router does not originate its TTZ router LSA or TTZ network LSAs. It will discover its TTZ neighbors.

Secondly, after configuring the TTZ, users may issue a CLI command on one router in the TTZ, which triggers every router in the TTZ to generate and distribute TTZ information among the routers in the TTZ. When the router receives the command, it originates its TTZ router LSA and TTZ network LSAs as needed, and distributes them to its TTZ neighbors. It also originates a TTZ control LSA with T=1 (indicating TTZ information generation and distribution for migration). When a router in the TTZ receives the LSA with T=1, it originates its TTZ router LSA and TTZ network LSAs as needed. In this stage, every router in the TTZ has dual roles. One is to function as a normal router. The other is to generate and distribute TTZ information.

Thirdly, users SHOULD check whether every router in the TTZ is ready for transferring to work as a TTZ router. A router in the TTZ is ready after it has received all the necessary information from all the routers in the TTZ. This information may be displayed on a router through a CLI command.

And then users may activate the TTZ through using a CLI command such as migrate to TTZ on one router in the TTZ. The router transfers to work as a TTZ router, generates and distributes a TTZ control LSA with M=1 (indicating Migrating to TTZ) after it receives the command.

After a router in the TTZ receives the TTZ control LSA with M=1, it

also transfers to work as a TTZ router. Thus, activating the TTZ on one TTZ router makes every router in the TTZ transfer to work as a TTZ router, which flushes its normal router LSA and network LSAs, computes routes through using the TTZ topology represented by TTZ LSAs and the topology outside of the TTZ.

For an edge router of the TTZ, transferring to work as a TTZ router comprises generating a router LSA to virtualize the TTZ and flooding this LSA to all its neighboring routers.

10.3. Adding a Router into TTZ

When a non TTZ router (say R1) is connected via a P2P link to a TTZ router (say T1) working as TTZ and there is a normal adjacency between them over the link, a user can configure TTZ on two ends of the link to add R1 into the TTZ to which T1 belongs. They discover TTZ each other in the same way as described in section 7.1.

When a number of non TTZ routers are connected via a broadcast link to a TTZ router (say T1) working as TTZ and there are normal adjacencies among them, a user configures TTZ on the connection to the link on every router to add the non TTZ routers into the TTZ to which T1 belongs. The DR for the link "forms" TTZ adjacency with each of the other routers if all the routers have the same TTZ ID configured on the connections to the link.

When a router (say R1) is connected via a P2P link to a TTZ router (say T1) and there is not any adjacency between them over the link, a user can configure TTZ on two ends of the link to add R1 into the TTZ to which T1 belongs. R1 and T1 will form an adjacency in the same way as described in section 7.2.

When a router (say R1) is connected via a broadcast link to a group of TTZ routers on the link and there is not any adjacency between R1 and any over the link, a user can configure TTZ on the connection to the link on R1 to add R1 into the TTZ to which the TTZ routers belong. R1 starts to form an adjacency with the DR for the link after the configuration.

11. Prototype Implementation

11.1. What are Implemented and Tested

1. CLI Commands for TTZ

The CLIs implemented and tested include:

- o the CLIs of the simpler option for configuring TTZ, and
- o the CLIs for controlling migration to TTZ.

2. Extensions to OSPF Protocols for TTZ

All the extensions defined in section "Extensions to OSPF Protocols" are implemented and tested except for rolling back from TTZ. The testing results illustrate:

- o A TTZ is virtualized to outside as its edge routers fully connected. Any router outside of the TTZ sees the edge routers (as normal routers) connecting each other and to some other routers.
- o The link state information about the routers and links inside the TTZ is contained within the TTZ. It is not distributed to any router outside of the TTZ.
- o TTZ is transparent. From a router inside a TTZ, it sees the topology (link state) outside of the TTZ. From a router outside of the TTZ, it sees the topology beyond the TTZ. The link state information outside of the TTZ is distributed through the TTZ.
- o TTZ is backward compatible. Any router outside of a TTZ does not need to support or know TTZ.

3. Smooth Migration to TTZ

The procedures and related protocol extensions for smooth migration to TTZ are implemented and tested. The testing results show:

- o A part of an area is smoothly migrated to a TTZ without any routing disruptions. The routes on every router are stable while the part of the area is being migrated to the TTZ.
- o Migration to TTZ is very easy to operate.

4. Add a Router to TTZ

Adding a router into TTZ is implemented and tested. The testing results illustrate:

- o A router can be easily added into a TTZ and becomes a TTZ router.

- o The router added into the TTZ is not seen on any router outside of the TTZ, but it is a part of the TTZ.

5. Leak TTZ Loopbacks Outside

Leaking loopback addresses in a TTZ to routers outside of the TTZ is implemented and tested. The testing results illustrate:

- o The loopback addresses inside the TTZ are distributed to the routers outside of the TTZ.
- o The loopback addresses are accessible from a router outside of the TTZ.

11.2. Implementation Experience

The implementation of TTZ is relatively easy compared to other features of OSPF. Re-using the existing OSPF code along with additional simple logic does the work. A couple of engineers started to work on implementing the TTZ from the middle of June, 2014 and finished coding it just before IETF 90. After some testing and bug fixes, it works as expected.

In our implementation, the link state information in a TTZ opaque LSA is stored in the same link state database as the link state information in a normal LSA. For each TTZ link in the TTZ opaque LSA stored, there is an additional flag, which is used to differentiate between a TTZ link and a Normal link.

Before migration to TTZ, every router in the TTZ computes its routing table using the normal links. After migration to TTZ, every router in the TTZ computes its routing table using the TTZ links and normal links. In the case that there are one TTZ link and one normal link to select, the TTZ link is used. In SPF calculation, the back-link check passes if and only if the corresponding new additional bit matches. If link type bit is TTZ link, then the lookup is for corresponding TTZ LSA. In case of normal link, the lookup is based on normal link.

12. Security Considerations

The mechanism described in this document does not raise any new security issues for the OSPF protocols.

13. IANA Considerations

TBD

14. Contributors

Veerendranatha Reddy Vallem
Huawei Technologies
Bangalore
India
Email: veerendranatharv@huawei.com

15. Acknowledgement

The author would like to thank Acee Lindem, Abhay Roy, Dean Cheng, Russ White, William McCall, Tony Przygienda, Lin Han and Yang Yu for their valuable comments on this draft.

16. References

16.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.
- [RFC2370] Coltun, R., "The OSPF Opaque LSA Option", RFC 2370, July 1998.
- [RFC5613] Zinin, A., Roy, A., Nguyen, L., Friedman, B., and D. Yeung, "OSPF Link-Local Signaling", RFC 5613, August 2009.

16.2. Informative References

- [RFC5441] Vasseur, JP., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, April 2009.

[RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.

Authors' Addresses

Huaimo Chen
Huawei Technologies
Boston, MA
USA

Email: huaimo.chen@huawei.com

Renwei Li
Huawei Technologies
2330 Central expressway
Santa Clara, CA
USA

Email: renwei.li@huawei.com

Anil Kumar S N
Huawei Technologies
Bangalore
India

Email: anil.sn@huawei.com

Gregory Cauchie
FRANCE

Email: greg.cauchie@gmail.com

Alvaro Retana
Cisco Systems, Inc.
7025 Kit Creek Rd.
Raleigh, NC 27709
USA

Email: aretana@cisco.com

Ning So
Tata Communications
2613 Fairbourne Cir.
Plano, TX 75082
USA

Email: ningso01@gmail.com

Vic Liu
China Mobile
No.32 Xuanwumen West Street, Xicheng District
Beijing, 100053
China

Email: liuzhiheng@chinamobile.com

Mehmet Toy
Comcast
1800 Bishops Gate Blvd.
Mount Laurel, NJ 08054
USA

Email: mehmet_toy@cable.comcast.com

Lei Liu
UC Davis
CA
USA

Email: liulei.kddi@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

U. Chunduri
Ericsson Inc.
X. Xu
Huawei
L. Contreras
Telefonica I+D
M. Boucadair
France Telecom
March 9, 2015

Using Self-defined Sub-TLVs for Agile Service Deployment
draft-chunduri-ospf-self-defined-sub-tlvs-03

Abstract

This document proposes a TLV within the body of the OSPF Router Information (RI) Opaque LSA, called Self-defined Sub-TLV Container TLV. Here the term OSPF means both OSPFv2 and OSPFv3. This attribute is meant to accommodate policy-based and deployment-specific use cases.

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 September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Sample Use Cases	3
3. Terminology	4
4. Self-defined Sub-TLV Container TLV	4
5. Self-defined Sub-TLV	5
6. Acknowledgements	6
7. IANA Considerations	6
8. Security Considerations	6
9. References	6
9.1. Normative References	6
9.2. Informative References	6
Authors' Addresses	7

1. Introduction

There are some use cases where OSPF is used for service auto-discovery by using node administrative tags [I-D.ietf-ospf-node-admin-tag] . One major benefit of using administrative tags rather than IANA defined TLVs or sub-TLVs to indicate different services is to facilitate the rapid deployment of new services without any need for the standardization of those TLVs or sub-TLVs. However, there are some special use cases where the service to be advertised has one or more attributes which need to be advertised as well. In such case, the administrative tag is not much applicable anymore.

To inherit the benefit of administrative tags (i.e., allowing operators to use OSPF for service auto-discovery without the need of any standardization process) while meeting the requirement of advertising services and their associated attributes, this document proposes a TLV within the body of the OSPF Router Information (RI) Opaque LSA, called Self-defined Sub-TLV Container TLV. With such TLV, operators could flexibly define one or more sub-TLVs indicating one or more services and their associated attributes without relying on any standardization process.

The characterization of the TLV and its associated sub-TLVs is local to the each administrative domain. Defining new sub-TLVs is therefore deployment-specific and policy-based. OSPF denotes both OSPFv2 and OSPFv3.

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. Sample Use Cases

There can be several possible use cases and applications for Self-defined Sub-TLV Container TLV defined in Section 4. This section provides few examples how operators can deploy services rapidly by advertising associated attributes. However, the illustrations listed below are not meant to be restrictive or exhaustive.

o Advertising Service Functions and it's attributes

Service Function nodes implementing various service functions within the network need to advertise each service function they are offering so that a control and/or management entity can decide which instance to invoke for the delivery of an added-value service or to react to particular events (such as failure of a service function instance). Each service can be identified by a dedicated sub-TLV type while the associated attributes/identifiers of the service are indicated by the value part of the corresponding sub-TLV. These identifiers MAY not be globally unique and MAY not be exposed outside of a given administrative domain. The Self-defined sub-TLV Container TLV could appear multiple times within a given Router Information (RI) Opaque LSA, when more than one service function instances needs to be advertised by a given node based on a local policy.

Advertising service functions and it's attributes also allow the controller to adjust its policies and react dynamically. Typical actions would be, to withdraw a service instance from being invoked in the context of a service delivery, update load balancing policies, dynamically activate a backup instance, etc.

The mechanisms, on how service information and attributes are used by an external controller (for example to steer the traffic) is beyond the scope of this document.

o Dissemination of dynamic information

It's possible for operators to disseminate the node local information like energy efficiency, congestion information, certain critical node statistics periodically to an external controller managing the network. How a Controller uses this information is beyond the scope of this document.

5. Self-defined Sub-TLV

The self-defined sub-TLV has the following structure and can be part of the Container TLV as defined in Section 4 within the body of the OSPF RI LSA.

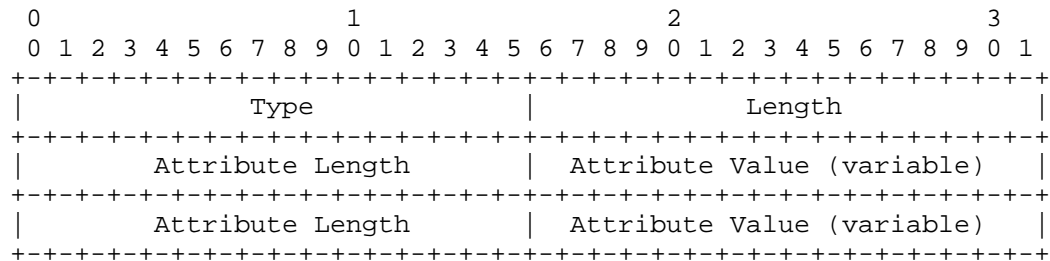


Figure 2: Self-defined Sub-TLV

Type: Per Operator/Local Policy.

Length: A 16-bit field that indicates the length of the value portion in octets and will be padded/formatted as described in Section 2.1 of [RFC4970].

Value: Represents the associated attribute of the service or Type defined locally (i.e., within a single administrative domain). The Value field contains one or more {Attribute-Len, Attribute-value} tuple. Attribute Length is of 2 bytes, for fixed formatting and Attribute value as represented by attribute length.

The meaning of the self-defined sub-TLV is totally opaque to OSPF.

Routers advertising the self-defined sub-TLV are configured to do so without knowing (or even explicitly supporting) functionality implied by the sub-TLV.

The meaning of a self-defined sub-TLV is defined by the network local policy and is controlled via configuration.

How a receiving node communicates the self-defined sub-TLVs with the policy manager is outside the scope of this document.

There is no need for any specification to define any self-defined sub-TLV. Furthermore, the semantics of the self-defined sub-TLV order has no meaning. That is, there is no implied meaning to the ordering of the self-defined sub-TLV that indicates a certain operation or set of operations that need to be performed based on the

ordering. The ordering of self-defined sub-TLVs and the interpretation of the self-defined sub-TLV is deployment-specific. Routers can be configured with local policies if the order of sub-TLV must be preserved. How a router is configured with additional instructions (such as order preservation) is implementation-specific.

6. Acknowledgements

Authors would like to thank Acee Lindem for reviewing and providing suggestions on the initial version of the document. Also thankful to Anton Smirnov, Peter Psenak and Les Ginsberg for their review and comments.

7. IANA Considerations

This document includes a request to IANA to allocate a TLV type code for the new RI LSA TLV proposed in Section 4 of this document from OSPF Router Information (RI) TLVs Registry defined by [RFC4970].

8. Security Considerations

This document does not introduce any new security risk other than what is specified by [RFC4970].

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.
- [RFC5838] Lindem, A., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, April 2010.

9.2. Informative References

- [I-D.ietf-ospf-node-admin-tag] Hegde, S., Raghuvver, H., Gredler, H., Shakir, R., Smirnov, A., Li, Z., and B. Decraene, "Advertising per-node administrative tags in OSPF", draft-ietf-ospf-node-admin-tag-00 (work in progress), October 2014.

Authors' Addresses

Uma Chunduri
Ericsson Inc.
300 Holger Way,
San Jose, California 95134
USA

Phone: 408 750-5678
Email: uma.chunduri@ericsson.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: <http://people.tid.es/LuisM.Contreras/>

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Open Shortest Path First IGP
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2014

S. Hegde
H. Raghuveer
H. Gredler
Juniper Networks, Inc.
R. Shakir
British Telecom
A. Smirnov
Cisco Systems, Inc.
Z. Li
Huawei Technologies
June 26, 2014

Advertising per-node administrative tags in OSPF
draft-hegde-ospf-node-admin-tag-02

Abstract

This document describes an extension to OSPF protocol [RFC2328] to add an optional operational capability, that allows tagging and grouping of the nodes in an OSPF domain. This allows simplification, ease of management and control over route and path selection based on configured policies.

This document describes the protocol extensions to disseminate per-node admin-tags to the OSPFv2 and OSPFv3 protocol.

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 <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 December 28, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Applicability	2
3. Administrative Tag TLV	3
4. OSPF per-node administrative tag TLV	3
4.1. TLV format	3
4.2. Elements of procedure	4
5. Applications	5
6. Security Considerations	8
7. IANA Considerations	9
8. Acknowledgments	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

This document provides mechanisms to advertise per-node administrative tags in the OSPF Router Information LSA [RFC4970]. In certain path-selection applications like for example in traffic-engineering or LFA backup selection there is a need to tag the nodes based on their roles in the network and have policies to prefer or prune a certain group of nodes.

2. Applicability

For the purpose of advertising per-node administrative tags within OSPF a new TLV is proposed. Because path selection is a functional

set which applies both to TE and non-TE applications, this new TLV is carried in the Router Information LSA (RI LSA) [RFC4970]

3. Administrative Tag TLV

An administrative Tag is a 32-bit integer value that can be used to identify a group of nodes in the OSPF domain.

The new TLV defined will be carried within an RI LSA for OSPFV2 and OSPFV3. Router information LSA [RFC4970] can have link, area or AS level flooding scope. Choosing the flooding scope to flood the group tags are defined by the policies and is a local matter.

The TLV specifies one or more administrative tag values. An OSPF node advertises the set of groups it is part of in the OSPF domain. (for example, all PE-nodes are configured with certain tag value, all P-nodes are configured with a different tag value in a domain). The total number of admin tags that a given router can advertise at one time is restricted to 64. If more tags are needed in future, multi-instantiating of the RI LSA [RFC4970] may be required.

4. OSPF per-node administrative tag TLV

4.1. TLV format

The format of the TLVs within the body of an RI LSA is the same as the format used by the Traffic Engineering Extensions to OSPF [RFC3630].

The LSA payload consists of one or more nested Type/Length/Value (TLV) triplets. The format of each TLV is:

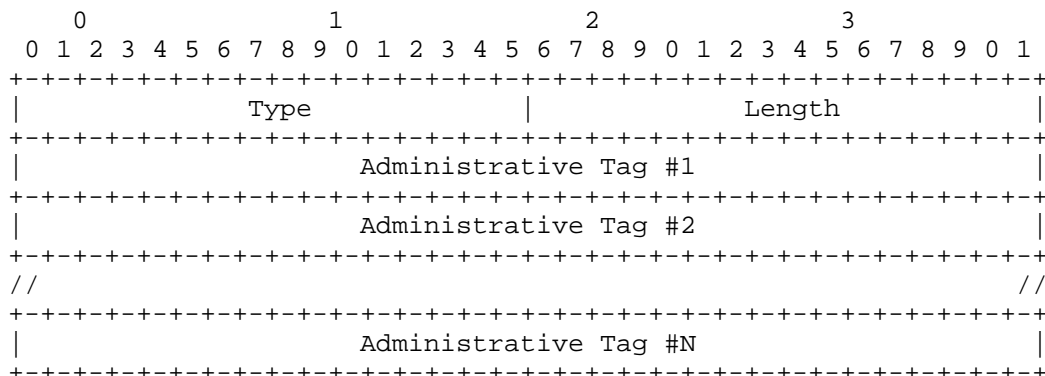


Figure 1: OSPF per-node Administrative Tag TLV

Type : TBA

Length: A 16-bit field that indicates the length of the value portion in octets and will be a multiple of 4 octets dependent on the number of tags advertised.

Value: A sequence of multiple 4 octets defining the administrative tags. The number of tags carried in this TLV is restricted to 64.

4.2. Elements of procedure

Meaning of the Node administrative tags is generally opaque to OSPF. Router advertising the Node administrative tag (or tags) may be configured to do so without knowing (or even explicitly supporting) functionality implied by the tag.

Interpretation of the tag values is implementation-specific. The meaning of a Node administrative tag is defined by the network local policy and is controlled via the configuration. There are no tag values defined by this specification.

The semantics of the tag order has no meaning. That is, there is no implied meaning to the ordering of the tags that indicates a certain operation or set of operations that need to be performed based on the ordering.

Each tag SHOULD be treated as an independent identifier that MAY be used in policy to perform a policy action. Whether or not tag A precedes or succeeds tag B SHOULD not change the meaning of the tag set.

To avoid incomplete or inconsistent interpretations of the Node administrative tags the same tag value MUST NOT be advertised by a router in RI LSAs of different scopes. The same tag MAY be advertised in multiple RI LSAs of the same scope, for example, OSPF Area Border Router (ABR) may advertise the same tag in area-scope RI LSAs in multiple areas connected to the ABR.

The Node administrative tags are not meant to be extended by the future OSPF standards. The new OSPF extensions MUST NOT require use of Node administrative tags or define well-known tag values. Instead, the future OSPF extensions must define their own data signaling tailored to the needs of the feature.

Being part of the RI LSA, the Node administrative tag TLV must be reasonably small and stable. In particular, but not limited to, implementations supporting the Node administrative tags MUST NOT tie advertised tags to changes in the network topology (both within and outside the OSPF domain) or reachability of routes.

5. Applications

This section lists several examples of how implementations might use the Node administrative tags. These examples are given only to demonstrate generic usefulness of the router tagging mechanism. Implementation supporting this specification is not required to implement any of the use cases. It is also worth noting that in some described use cases routers configured to advertise tags help other routers in their calculations but do not themselves implement the same functionality.

1. Service auto-discovery

Router tagging may be used to automatically discover group of routers sharing a particular service.

For example, service provider might desire to establish full mesh of MPLS TE tunnels between all PE routers in the area of MPLS VPN network. Marking all PE routers with a tag and configuring devices with a policy to create MPLS TE tunnels to all other devices advertising this tag will automate maintenance of the full mesh. When new PE router is added to the area, all other PE devices will open TE tunnels to it without the need of reconfiguring them.

2. Fast-Rerouting policy

Increased deployment of Loop Free Alternates (LFA) as defined in [RFC5286] poses operation and management challenges.

[I-D.litkowski-rtgwg-lfa-manageability] proposes policies which, when implemented, will ease LFA operation concerns.

One of the proposed refinements is to be able to group the nodes in IGP domain with administrative tags and engineer the LFA based on configured policies.

(a) Administrative limitation of LFA scope

Service provider access infrastructure is frequently designed in layered approach with each layer of devices serving different purposes and thus having different hardware capabilities and configured software features. When LFA repair paths are being computed, it may be desirable to exclude devices from being considered as LFA candidates based on their layer.

For example, if the access infrastructure is divided into the Access, Distribution and Core layers it may be desirable for a Distribution device to compute LFA only via Distribution or Core devices but not via Access devices. This may be due to features enabled on Access routers; due to capacity limitations or due to the security requirements. Managing such a policy via configuration of the router computing LFA is cumbersome and error prone.

With the Node administrative tags it is possible to assign a tag to each layer and implement LFA policy of computing LFA repair paths only via neighbors which advertise the Core or Distribution tag. This requires minimal per-node configuration and network automatically adapts when new links or routers are added.

(b) LFA calculation optimization

Calculation of LFA paths may require significant resources of the router. One execution of Dijkstra algorithm is required for each neighbor eligible to become next hop of repair paths. Thus a router with a few hundreds of neighbors may need to execute the algorithm hundreds of times before the best (or even valid) repair path is found. Manually excluding from the calculation neighbors which are known to provide no valid LFA (such as single-connected routers) may significantly reduce number of Dijkstra algorithm runs.

LFA calculation policy may be configured so that routers advertising certain tag value are excluded from LFA calculation even if they are otherwise suitable.

3. Controlling Remote LFA tunnel termination

[I-D.ietf-rtgwg-remote-lfa] proposed method of tunneling traffic after connected link failure to extend the basic LFA coverage and algorithm to find tunnel tail-end routers fitting LFA requirement. In most cases proposed algorithm finds more than one candidate tail-end router. In real life network it may be desirable to exclude some nodes from the list of candidates based on the local policy. This may be either due to known limitations of the node (the router does not accept targeted LDP sessions required to implement Remote LFA tunneling) or due to administrative requirements (for example, it may be desirable to choose tail-end router among co-located devices).

The Node administrative tag delivers simple and scalable solution. Remote LFA can be configured with a policy to accept during the tail-end router calculation as candidates only routers advertising certain tag. Tagging routers allows to both exclude nodes not capable of serving as Remote LFA tunnel tail-ends and to define a region from which tail-end router must be selected.

4. Mobile backhaul network service deployment

The topology of mobile backhaul network usually adopts ring topology to save fiber resource and it is divided into the aggregate network and the access network. Cell Site Gateways(CSGs) connects the eNodeBs and RNC(Radio Network Controller) Site Gateways(RSGs) connects the RNCs. The mobile traffic is transported from CSGs to RSGs. The network takes a typical aggregate traffic model that more than one access rings will attach to one pair of aggregate site gateways(ASGs) and more than one aggregate rings will attach to one pair of RSGs.

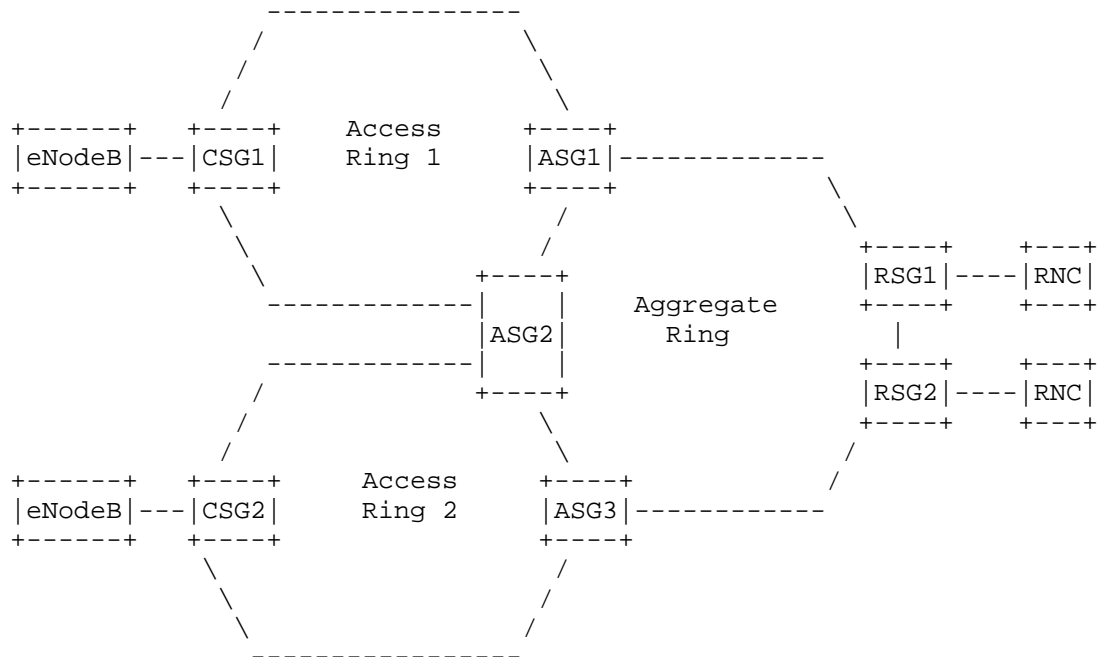


Figure 2: Mobile Backhaul Network

A typical mobile backhaul network with access rings and aggregate links is shown in figure above. The mobile backhaul networks deploy traffic engineering due to the strict Service Level Agreements(SLA). The TE paths may have additional constraints to avoid passing via different access rings or to get completely disjoint backup TE paths. The mobile backhaul networks towards the access side change frequently due to the growing mobile traffic and addition of new eNodeBs. It's complex to satisfy the requirements using cost, link color or explicit path configurations. The node administrative tag defined in this document can be effectively used to solve the problem for mobile backhaul networks. The nodes in different rings can be assigned with specific tags. TE path computation can be enhanced to consider additional constraints based on node administrative tags.

6. Security Considerations

This document does not introduce any further security issues other than those discussed in [RFC2328] and [RFC5340].

7. IANA Considerations

IANA maintains the registry for the TLVs. OSPF Administrative Tags will require one new type code for the TLV defined in this document.

8. Acknowledgments

Thanks to Bharath R and Pushpasis Sarakar for useful inputs. Thanks to Chris Bowers for providing useful inputs to remove ambiguity related to tag-ordering.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.
- [RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

9.2. Informative References

- [I-D.ietf-rtgwg-remote-lfa]
Bryant, S., Filsfils, C., Previdi, S., Shand, M., and S. Ning, "Remote LFA FRR", draft-ietf-rtgwg-remote-lfa-02 (work in progress), May 2013.
- [I-D.litkowski-rtgwg-lfa-manageability]
Litkowski, S., Decraene, B., Filsfils, C., and K. Raza, "Operational management of Loop Free Alternates", draft-litkowski-rtgwg-lfa-manageability-01 (work in progress), February 2013.
- [RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, September 2008.

Authors' Addresses

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

Email: shraddha@juniper.net

Harish Raghuveer
Juniper Networks, Inc.
Embassy Business Park
Bangalore 560093
India

Email: hraghuveer@juniper.net

Hannes Gredler
Juniper Networks, Inc.
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: hannes@juniper.net

Rob Shakir
British Telecom

Email: rob.shakir@bt.com

Anton Smirnov
Cisco Systems, Inc.
De Kleetlaan 6a
Diegem 1831
Belgium

Email: as@cisco.com

Li Zhenbin
Huawei Technologies
Huawei Bld. No.156 Beiqing Rd
Beijing 100095
China

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 21, 2016

P. Psenak
Cisco Systems
H. Gredler
Juniper Networks, Inc.
R. Shakir
Individual Contributor
W. Henderickx
Alcatel-Lucent
J. Tantsura
Ericsson
A. Lindem
Cisco Systems
August 20, 2015

OSPFv2 Prefix/Link Attribute Advertisement
draft-ietf-ospf-prefix-link-attr-13.txt

Abstract

OSPFv2 requires functional extension beyond what can readily be done with the fixed-format Link State Advertisements (LSAs) as described in RFC 2328. This document defines OSPF opaque LSAs based on Type-Length-Value (TLV) tuples that can be used to associate additional attributes with prefixes or links. Dependent on the application, these prefixes and links may or not be advertised in the fixed-format LSAs. The OSPF opaque LSAs are optional and fully backward compatible.

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 February 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements notation	3
2. OSPFv2 Extended Prefix Opaque LSA	3
2.1. OSPFv2 Extended Prefix TLV	5
3. OSPFv2 Extended Link Opaque LSA	8
3.1. OSPFv2 Extended Link TLV	9
4. Backward Compatibility	10
5. Implementation Status	10
5.1. Implementation Survey Results	11
6. Security Considerations	12
7. IANA Considerations	12
7.1. OSPF Extended Prefix Opaque LSA TLV Registry	13
7.2. OSPF Extended Prefix TLV Sub-TLV Registry	13
7.3. OSPF Extended Prefix TLV Flags Registry	13
7.4. OSPF Extended Link Opaque LSA TLV Registry	14
7.5. OSPF Extended Link TLV Sub-TLV Registry	14
8. Acknowledgments	14
9. References	15
9.1. Normative References	15

9.2. Informative References	15
Authors' Addresses	16

1. Introduction

OSPFv2 requires functional extension beyond what can readily be done with the fixed-format Link State Advertisements (LSAs) as described in RFC 2328 [OSPFV2]. This document defines OSPF opaque LSAs based on Type-Length-Value (TLV) tuples that can be used to associate additional attributes with prefixes or links. Dependent on the application, these prefixes and links may or not be advertised in the fixed-format LSAs. The OSPF opaque LSAs are optional and fully backward compatible. This is in contrast to the approach taken in OSPFv3 [I-D.ietf-ospf-ospfv3-lsa-extend] where the existing LSAs will be replaced by TLV-based extended LSAs.

New requirements such as source/destination routing, route tagging, and segment routing necessitate this extension.

This specification defines the following OSPFv2 opaque LSAs:

1. OSPFv2 Extended Prefix Opaque LSA - Allows advertisement of additional attributes for prefixes advertised in Router-LSAs, Network-LSAs, Network-Summary-LSAs, NSSA-LSAs, and AS-External-LSAs [OSPFV2]
2. OSPFv2 Extended Link Opaque LSA - Allows advertisement of additional attributes for links advertised in Router-LSAs.

Additionally, the following TLVs are defined:

1. OSPFv2 Extended Prefix TLV - Top-level TLV advertising attributes for a prefix in the OSPFv2 Extended Prefix Opaque LSA.
2. OSPFv2 Extended Link TLV - Top-level TLV advertising attributes for a link in the OSPFv2 Extended Link Opaque LSA.

1.1. Requirements notation

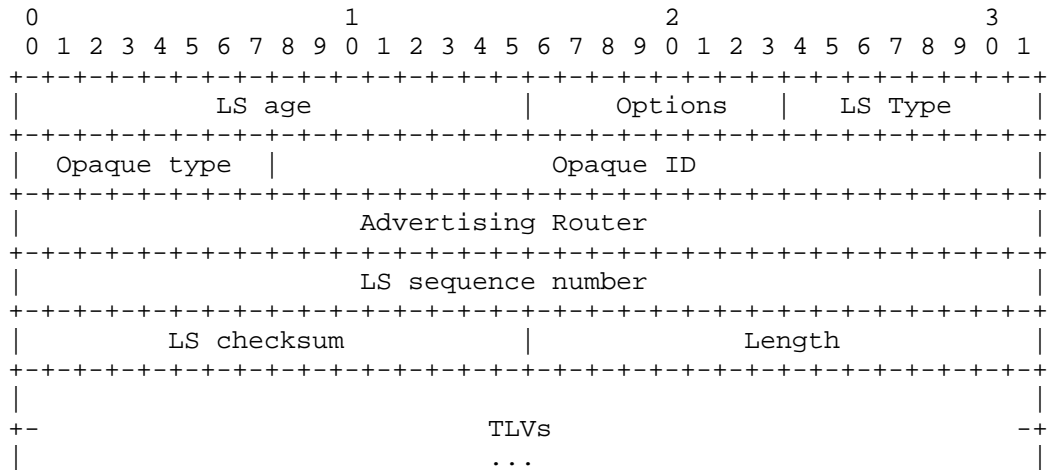
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-KEYWORDS].

2. OSPFv2 Extended Prefix Opaque LSA

The OSPFv2 Extended Prefix Opaque LSA will be used to advertise additional prefix attributes. Opaque LSAs are described in [OPAQUE].

Multiple OSPFv2 Extended Prefix Opaque LSAs can be advertised by an OSPFv2 router. The flooding scope of the OSPFv2 Extended Prefix Opaque LSA depends on the scope of the advertised prefixes and is under the control of the advertising router. In some cases (e.g., mapping server deployment [SEGMENT-ROUTING]), the LSA flooding scope may be greater than the scope of the corresponding prefixes.

The format of the OSPFv2 Extended Prefix Opaque LSA is as follows:



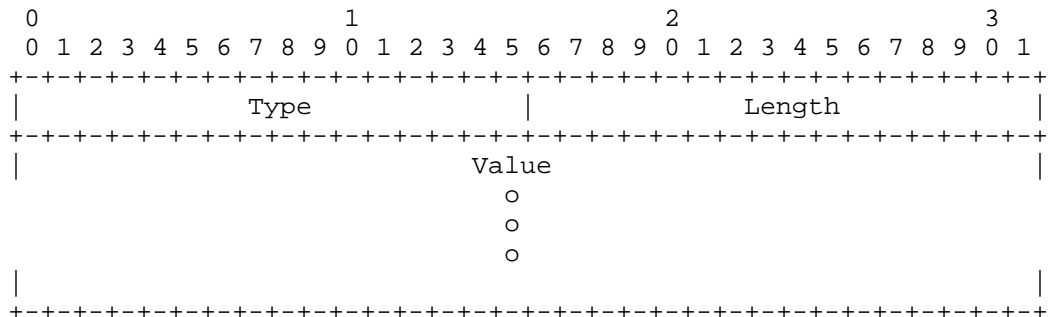
OSPFv2 Extended Prefix Opaque LSA

The opaque type used by OSPFv2 Extended Prefix Opaque LSA is 7. The opaque type is used to differentiate the various type of OSPFv2 Opaque LSA and is described in section 3 of [OPAQUE]. The LS Type may be 10 or 11 indicating that the Opaque LSA flooding scope is area-local (10) or AS-wide (11) [OPAQUE]. The LSA "Length" field [OSPFV2] 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 Extended Prefix Opaque LSAs. For OSPFv2 Extended Prefix Opaque LSAs, the Opaque ID has no semantic significance other than to differentiate Extended Prefix Opaque LSAs originated by the same OSPFv2 router. If multiple Extended Prefix Opaque LSAs include the same prefix, the attributes from the Opaque LSA with the lowest Opaque ID SHOULD be used.

The format of the TLVs within the body of the OSPFv2 Extended Prefix Opaque LSA is the same as the format used by the Traffic Engineering Extensions to OSPF [TE]. The variable TLV section consists of one or

more nested Type/Length/Value (TLV) tuples. Nested TLVs are also referred to as sub-TLVs. The format of each TLV is:

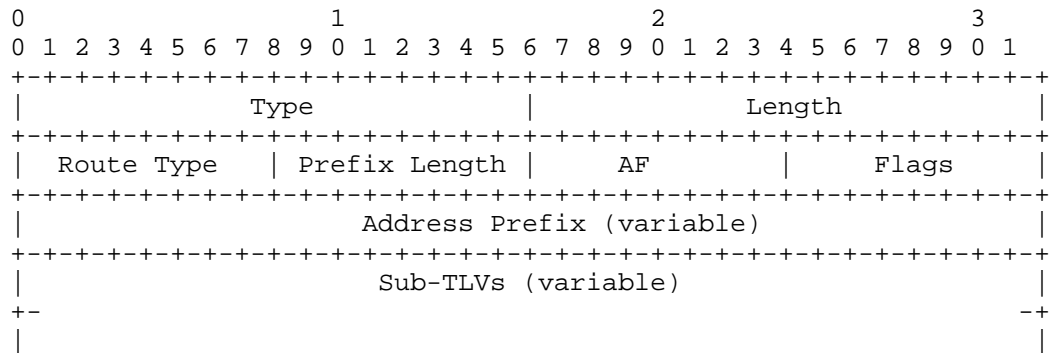


TLV Format

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-byte 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.

2.1. OSPFv2 Extended Prefix TLV

The OSPF Extended Prefix TLV is used to advertise additional attributes associated with the prefix. Multiple OSPF Extended Prefix TLVs MAY be advertised in each OSPFv2 Extended Prefix Opaque LSA. However, since the opaque LSA type defines the flooding scope, the LSA flooding scope MUST satisfy the application specific requirements for all the prefixes included in a single OSPFv2 Extended Prefix Opaque LSA. The OSPF Extended Prefix TLV has the following format:



OSPFv2 Extended Prefix TLV

Type

The TLV type. The value is 1 for this TLV type.

Length

Variable dependent on sub-TLVs.

Route Type

Route type: type of the OSPF route. If the route type is 0 (Unspecified), the information inside the OSPF External Prefix TLV applies to the prefix regardless of prefix's route-type. This is useful when prefix specific attributes are advertised by an external entity that is not aware of the route-type associated with the prefix. Supported types are:

- 0 - Unspecified
- 1 - Intra-Area
- 3 - Inter-Area
- 5 - AS External
- 7 - NSSA External

These route types correspond directly to the OSPFv2 LSAs types as defined in <http://www.iana.org/assignments/ospfv2-parameters/ospfv2-parameters.xhtml#ospfv2-parameters-5>. Specification of route types other than those defined will prevent correlation with existing OSPFv2 LSAs and is beyond the scope this specification.

Prefix Length

Length in the prefix in bits.

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.

Flags

This one octet field contains flags applicable to the prefix. Supported Flags include:

0x80 - A-Flag (Attach flag): An Area Border Router (ABR) generating an Extended Prefix TLV for inter-area prefix that is locally connected or attached in other connected area SHOULD set this flag.

0x40 - N-Flag (Node Flag): Set when the prefix identifies the advertising router i.e., the prefix is a host prefix advertising a globally reachable address typically associated with a loopback address. The advertising router MAY choose to not set this flag even when the above conditions are met. If the flag is set and the prefix length is not a host prefix then the flag MUST be ignored. The flag is preserved when the OSPFv2 Extended Prefix Opaque LSA is propagated between areas.

Address Prefix

For the address family IPv4 unicast, the prefix itself encoded as a 32-bit value. The default route is represented by a prefix of length 0. Prefix encoding for other address families is beyond the scope of this specification.

If this TLV is advertised multiple times for the same prefix in the same OSPFv2 Extended Prefix Opaque LSA, only the first instance of the TLV is used by receiving OSPFv2 Routers. This situation SHOULD be logged as an error.

If this TLV is advertised multiple times for the same prefix in different OSPFv2 Extended Prefix Opaque LSAs originated by the same OSPF router, the OSPF advertising router is re-originating Extended Prefix Opaque LSAs for multiple prefixes and is most likely repacking Extended-Prefix-TLVs in Extended Prefix Opaque LSAs. In this case, the Extended-Prefix-TLV in the Extended Prefix Opaque LSA with the smallest Opaque ID is used by receiving OSPFv2 Routers. This situation may be logged as a warning.

It is RECOMMENDED that OSPF routers advertising Extended Prefix TLVs in different Extended Prefix Opaque LSAs re-originate these LSAs in ascending order of Opaque ID to minimize the disruption.

If this TLV is advertised multiple times for the same prefix in different OSPFv2 Extended Prefix Opaque LSAs originated by different OSPF routers, the application using the information is required to determine which OSPFv2 Extended Prefix Opaque LSA is used. For example, the application could prefer the LSA providing the best path to the prefix.

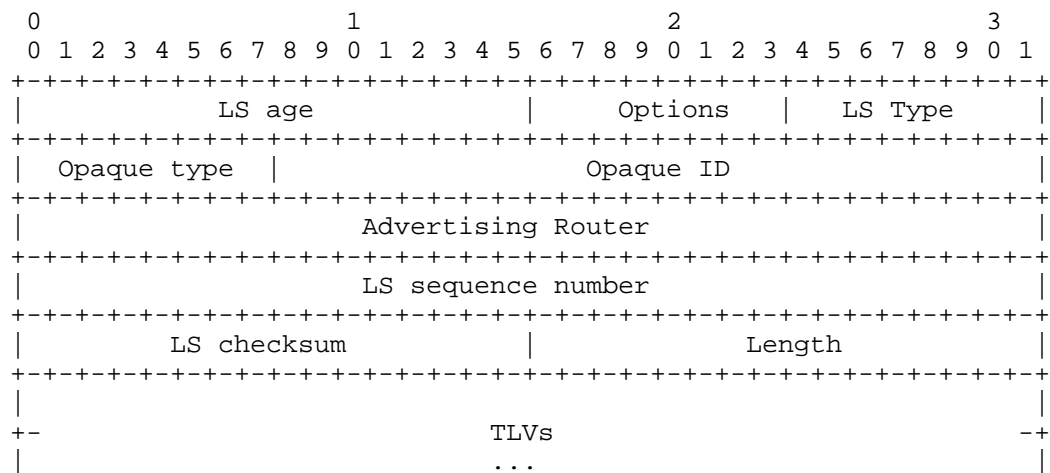
This document creates a registry for OSPF Extended Prefix sub-TLVs in Section 7.

3. OSPFv2 Extended Link Opaque LSA

The OSPFv2 Extended Link Opaque LSA will be used to advertise additional link attributes. Opaque LSAs are described in [OPAQUE].

The OSPFv2 Extended Link Opaque LSA has an area flooding scope. Multiple OSPFv2 Extended Link Opaque LSAs can be advertised by a single router in an area.

The format of the OSPFv2 Extended Link Opaque LSA is as follows:



OSPFv2 Extended Link Opaque LSA

The Opaque type used by OSPFv2 Extended Link Opaque LSA is 8. The LS Type is 10 indicating that the Opaque LSA flooding scope is area-local [OPAQUE]. The opaque type is used to differentiate the various type of OSPFv2 Opaque LSA and is described in section 3 of [OPAQUE]. The LSA "Length" field [OSPFV2] represents the total length (in octets) of the Opaque LSA including the LSA header and all TLVs (including padding).

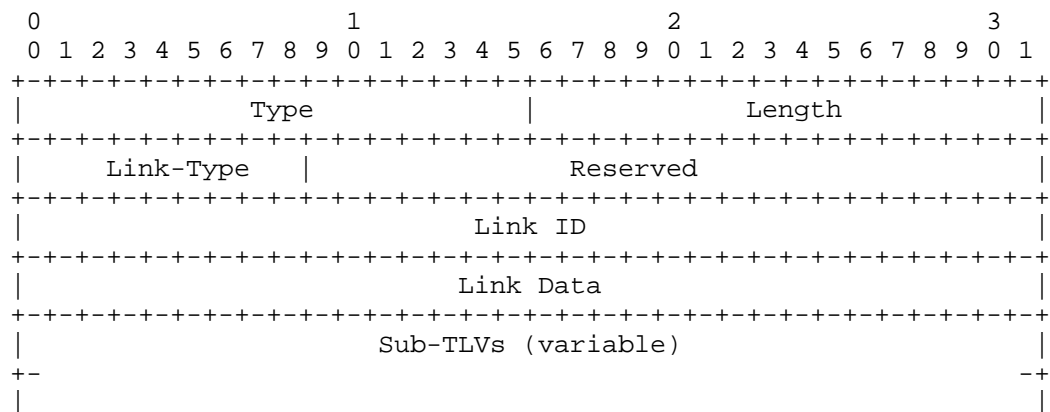
The Opaque ID field is an arbitrary value used to maintain multiple Extended Prefix Opaque LSAs. For OSPFv2 Extended Link Opaque LSAs, the Opaque ID has no semantic significance other than to differentiate Extended Link Opaque LSAs originated by the same OSPFv2 router. If multiple Extended Link Opaque LSAs include the same link, the attributes from the Opaque LSA with the lowest Opaque ID will be used.

The format of the TLVs within the body of the OSPFv2 Extended Link Opaque LSA is the same as described in Section 2.

3.1. OSPFv2 Extended Link TLV

The OSPFv2 Extended Link TLV is used to advertise various attributes of the link. It describes a single link and is constructed of a set of Sub-TLVs. There are no ordering requirements for the Sub-TLVs. Only one Extended Link TLV SHALL be advertised in each Extended Link Opaque LSA, allowing for fine granularity changes in the topology.

The Extended Link TLV has following format:



OSPFv2 Extended Link TLV

Type

The TLV type. The value is 1 for this TLV type.

Length

Variable dependent on sub-TLVs.

Link-Type

Link-Type is defined in section A.4.2 of [OSPFV2] and <http://www.iana.org/assignments/ospfv2-parameters/ospfv2-parameters.xhtml#ospfv2-parameters-6>. Specification of link types other than those defined will prevent correlation with existing OSPFv2 Router-LSA links and is beyond the scope this specification.

Link-ID

Link-ID is defined in section A.4.2 of [OSPFV2].

Link Data

Link-Data is defined in section A.4.2 of [OSPFV2].

If this TLV is advertised multiple times in the same OSPFv2 Extended Link Opaque LSA, only the first instance of the TLV is used by receiving OSPFv2 Routers. This situation SHOULD be logged as an error.

If this TLV is advertised multiple times for the same link in different OSPFv2 Extended Link Opaque LSAs originated by the same OSPF router, the Extended Link TLV in the Extended Link Opaque LSA with the smallest Opaque ID is used by receiving OSPFv2 Routers. This situation may be logged as a warning.

It is RECOMMENDED that OSPF routers advertising Extended Link TLVs in different Extended Link Opaque LSAs re-originate these LSAs in ascending order of Opaque ID to minimize the disruption.

This document creates a registry for OSPF Extended Link sub-TLVs in Section 7.

4. Backward Compatibility

Since opaque OSPFv2 LSAs are optional and backward compatible [OPAQUE], the extensions described herein are fully backward compatible. However, future OSPFv2 applications utilizing these extensions MUST address backward compatibility of the corresponding functionality.

5. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982. The description of implementations in this section is intended to

assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. Implementation Survey Results

An implementation survey with seven questions related to the implementer's support of OSPFv2 Prefix/Link Attributes was sent to the OSPF WG list and several known implementers. This section contains responses from four implementers who completed the survey. No external means were used to verify the accuracy of the information submitted by the respondents. The respondents are considered experts on the products they reported on. Additionally, responses were omitted from implementers who indicated that they have not implemented the function yet.

Four vendors and one open source entity replied to the survey. These included Alcatel-Lucent, Cisco, Huawei, Juniper, and FreeRouter (<http://freerouter.nop.hu>). Cisco and Alcatel-Lucent also did interoperability testing. FreeRouter did interoperability testing with Cisco. The Cisco, Alcatel-Lucent, and FreeRouter implementations are in released software versions. The Huawei and Juniper implementation software releases are pending. For prefix attributes, the recent change incorporating the A-Flag is pending implementation for all four vendors. The FreeRouter implementation includes support for the A-Flag. Implementation of the N-flag is pending for the Huawei and Juniper implementations. Otherwise, all the survey respondents have full implementations. For all four vendors and the FreeRouter implementation, segment routing [SEGMENT-ROUTING] was an application making use of the extensions. Additionally, Cisco has implemented Topology-Independent Loop-Free Alternatives (TI-LFA) [TI-LFA] and Bit Indexed Egress Replication (BIER) advertisement [BIER].

Alcatel-Lucent's support of this specification is included in SR OS, Release 13.0.R4. Cisco's support is included in IOS-XR 5.3.2. The

FreeRouter implementation is available in the FreeRouter 15.6.4 distribution. Huawei and Juniper will respectively provide support in future versions Versatile Routing Platform (VRP) and Juniper Network Operating System (JUNOS).

6. Security Considerations

In general, new LSAs defined in this document are subject to the same security concerns as those described in [OSPFV2] and [OPAQUE].

OSPFv2 applications utilizing these OSPFv2 extensions must define the security considerations relating to those applications in the specifications corresponding to those applications.

Additionally, implementations must assure that malformed TLV and Sub-TLV permutations are detected and do not provide a vulnerability for attackers to crash the OSPFv2 router or routing process. Malformed LSAs MUST NOT be stored in the Link State Database (LSDB), acknowledged, or reflooded. Reception of malformed LSAs SHOULD be counted and/or logged for further analysis. In this context, a malformed LSA is one which cannot be parsed due to a TLV or Sub-TLV overrunning the end of the subsuming LSA, TLV, or sub-TLV or where there is data remaining to be parsed but the length of the remaining data is less than the size of a TLV header.

7. IANA Considerations

This specification updates the Opaque Link-State Advertisements (LSA) Option Types with the following values:

- o 7 (IANA Early Allocation [RFC7120]) - OSPFv2 Extended Prefix Opaque LSA
- o 8 (IANA Early Allocation [RFC7120]) - OSPFv2 Extended Link Opaque LSA

This specification also creates five new registries:

- o OSPF Extended Prefix Opaque LSA TLVs
- o OSPF Extended Prefix TLV Sub-TLVs
- o OSPF Extended Prefix TLV Flags
- o OSPF Extended Link Opaque LSA TLVs
- o OSPF Extended Link TLV Sub-TLVs

7.1. OSPF Extended Prefix Opaque LSA TLV Registry

The "OSPF Extend Prefix Opaque LSA TLV" registry will define top-level TLVs for the Extended Prefix Opaque LSAs and should be added to the "Open Shortest Path First v2 (OSPFv2) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

The following initial values are allocated:

- o 0 - Reserved
- o 1 - OSPF Extended Prefix TLV

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.

7.2. OSPF Extended Prefix TLV Sub-TLV Registry

The "OSPF Extended Prefix TLV sub-TLV" registry will define sub-TLVs at any level of nesting for Extended Prefix TLVs and should be added to the "Open Shortest Path First v2 (OSPFv2) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

The following initial values are allocated:

- o 0 - Reserved

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. OSPF Extended Prefix TLV Flags Registry

The "OSPF Extended Prefix TLV Flags" registry will define the bits in the 8-bit Extended Prefix TLV Flags (Section 2.1). This specification defines the N (0x80) and A (0x40) bits. The registry should be added to the "Open Shortest Path First v2 (OSPFv2) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

7.4. OSPF Extended Link Opaque LSA TLV Registry

The "OSPF Extended Link Opaque LSA TLV" registry will define top-level TLVs for Extended Link Opaque LSAs and should be added to the "Open Shortest Path First v2 (OSPFv2) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

Following initial values are allocated:

- o 0 - Reserved
- o 1 - OSPFv2 Extended Link TLV

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.

7.5. OSPF Extended Link TLV Sub-TLV Registry

The OSPF Extended Link TLV sub-TLV registry will define sub-TLVs at any level of nesting for Extended Link TLVs and should be added to the "Open Shortest Path First v2 (OSPFv2) Parameters" registries group. New values can be allocated via IETF Review or IESG Approval.

The following initial values are allocated:

- o 0 - Reserved

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.

8. Acknowledgments

We would like to thank Anton Smirnov for his contribution.

Thanks to Tony Przygienda for his review and comments.

Thanks to Wim Henderickx, Greg Harkins, Peter Psenak, Eric Wu, Shraddha Hegde, and Csaba Mate for their responses to the implementation survey.

Thanks to Tom Petch for review and comments.

Thanks to Alia Atlas and Alvaro Retana for AD review and comments.

Thanks to Carlos Pignataro and Ron Bonica for Operations Directorate review and comments.

Thanks to Suresh Krishnan for Gen-ART review and comments.

Thanks to Ben Campbell, Kathleen Moriarty, and Barry Leiba for IESG review and comments.

9. References

9.1. Normative References

- [OPAQUE] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.
- [OSPFV2] Moy, J., "OSPF Version 2", RFC 2328, April 1998.
- [RFC-KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.
- [TE] Katz, D., Yeung, D., and K. Kompella, "Traffic Engineering Extensions to OSPF", RFC 3630, September 2003.

9.2. Informative References

- [BIER] Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., Przygienda, T., Zhang, J., and S. Aldrin, "OSPF Extensions for BIER", draft-ietf-bier-ospf-bier-extensions-00.txt (work in progress), April 2015.
- [I-D.ietf-ospf-ospfv3-lsa-extend] Lindem, A., Mirtorabi, S., Roy, A., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-06 (work in progress), February 2015.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.
- [SEGMENT-ROUTING] 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-05.txt (work in progress), June 2015.

[TI-LFA] Francois, P., Filsfils, C., Bashandy, A., Decraene, B.,
and S. Litkowski, "Topology Independent Fast Reroute using
Segment Routing", draft-francois-rtgwg-segment-routing-ti-
lfa-00.txt (work in progress), August 2014.

Authors' Addresses

Peter Psenak
Cisco Systems
Apollo Business Center
Mlynske nivy 43
Bratislava, 821 09
Slovakia

Email: ppsenak@cisco.com

Hannes Gredler
Juniper Networks, Inc.
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: hannes@juniper.net

Rob Shakir
Individual Contributor
London
UK

Email: rjs@rob.sh

Wim Henderickx
Alcatel-Lucent
Copernicuslaan
Antwerp, 2018 94089
Belgium

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson
300 Holger Way
San Jose, CA 95134
USA

Email: jeff.tantsura@ericsson.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Open Shortest Path First IGP
Internet-Draft
Intended status: Standards Track
Expires: June 6, 2019

P. Psenak, Ed.
S. Previdi, Ed.
C. Filsfils
Cisco Systems, Inc.
H. Gredler
RtBrick Inc.
R. Shakir
Google, Inc.
W. Henderickx
Nokia
J. Tantsura
Apstra, Inc.
December 3, 2018

OSPF Extensions for Segment Routing
draft-ietf-ospf-segment-routing-extensions-27

Abstract

Segment Routing (SR) allows a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF).

This draft describes the OSPFv2 extensions required for Segment Routing.

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 June 6, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Segment Routing Identifiers	3
2.1. SID/Label Sub-TLV	4
3. Segment Routing Capabilities	4
3.1. SR-Algorithm TLV	4
3.2. SID/Label Range TLV	6
3.3. SR Local Block TLV	8
3.4. SRMS Preference TLV	10
4. OSPF Extended Prefix Range TLV	11
5. Prefix SID Sub-TLV	13
6. Adjacency Segment Identifier (Adj-SID)	16
6.1. Adj-SID Sub-TLV	17
6.2. LAN Adj-SID Sub-TLV	18
7. Elements of Procedure	19
7.1. Intra-area Segment routing in OSPFv2	19
7.2. Inter-area Segment routing in OSPFv2	20
7.3. Segment Routing for External Prefixes	21
7.4. Advertisement of Adj-SID	22
7.4.1. Advertisement of Adj-SID on Point-to-Point Links	22
7.4.2. Adjacency SID on Broadcast or NBMA Interfaces	22
8. IANA Considerations	22
8.1. OSPF Router Information (RI) TLVs Registry	22
8.2. OSPFv2 Extended Prefix Opaque LSA TLVs Registry	23
8.3. OSPFv2 Extended Prefix TLV Sub-TLVs Registry	23
8.4. OSPFv2 Extended Link TLV Sub-TLVs Registry	23
8.5. IGP Algorithm Type Registry	23
9. Implementation Status	24
10. Security Considerations	25
11. Contributors	26

12. Acknowledgements	26
13. References	26
13.1. Normative References	26
13.2. Informative References	27
Authors' Addresses	28

1. Introduction

Segment Routing (SR) allows a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF). Prefix segments represent an ECMP-aware shortest-path to a prefix (or a node), as per the state of the IGP topology. Adjacency segments represent a hop over a specific adjacency between two nodes in the IGP. A prefix segment is typically a multi-hop path while an adjacency segment, in most cases, is a one-hop path. SR's control-plane can be applied to both IPv6 and MPLS data-planes, and does not require any additional signalling (other than IGP extensions). The IPv6 data plane is out of the scope of this specification - it is not applicable to OSPFv2 which only supports the IPv4 address-family. When used in MPLS networks, SR paths do not require any LDP or RSVP-TE signalling. However, SR can interoperate in the presence of LSPs established with RSVP or LDP.

There are additional segment types, e.g., Binding SID defined in [I-D.ietf-spring-segment-routing].

This draft describes the OSPF extensions required for Segment Routing.

Segment Routing architecture is described in [I-D.ietf-spring-segment-routing].

Segment Routing use cases are described in [RFC7855].

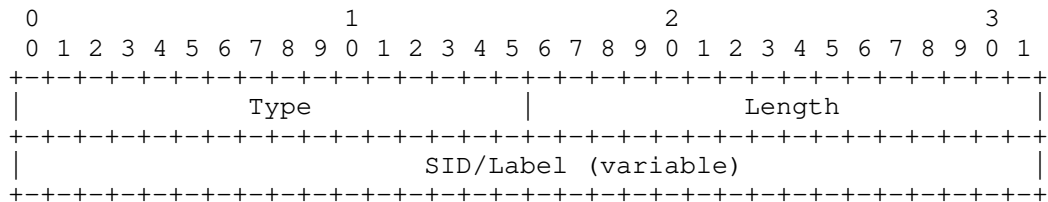
2. Segment Routing Identifiers

Segment Routing defines various types of Segment Identifiers (SIDs): Prefix-SID, Adjacency-SID, LAN Adjacency SID, and Binding SID.

Extended Prefix/Link Opaque LSAs defined in [RFC7684] are used for advertisements of the various SID types.

2.1. SID/Label Sub-TLV

The SID/Label Sub-TLV appears in multiple TLVs or Sub-TLVs defined later in this document. It is used to advertise the SID or label associated with a prefix or adjacency. The SID/Label Sub-TLV has following format:



where:

Type: 1

Length: Variable, 3 or 4 octet

SID/Label: If length is set to 3, then the 20 rightmost bits represent a label. If length is set to 4, then the value represents a 32-bit SID.

The receiving router MUST ignore the SID/Label Sub-TLV if the length is other than 3 or 4.

3. Segment Routing Capabilities

Segment Routing requires some additional router capabilities to be advertised to other routers in the area.

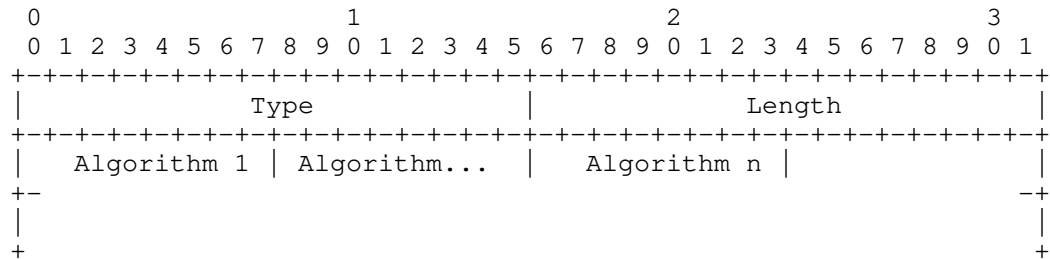
These SR capabilities are advertised in the Router Information Opaque LSA (defined in [RFC7770]). The TLVs defined below are applicable to both OSPFv2 and OSPFv3; see also [I-D.ietf-ospf-ospfv3-segment-routing-extensions]

3.1. SR-Algorithm TLV

The SR-Algorithm TLV is a top-level TLV of the Router Information Opaque LSA (defined in [RFC7770]).

The SR-Algorithm TLV is optional. It SHOULD only be advertised once in the Router Information Opaque LSA. If the SR-Algorithm TLV is not advertised by the node, such node is considered as not being segment routing capable.

An SR Router can use various algorithms when calculating reachability to OSPF routers or prefixes in an OSPF area. Examples of these algorithms are metric based Shortest Path First (SPF), various flavors of Constrained SPF, etc. The SR-Algorithm TLV allows a router to advertise the algorithms currently used by the router to other routers in an OSPF area. The SR-Algorithm TLV has following format:



where:

Type: 8

Variable, in octets, dependent on number of algorithms advertised.

Algorithm: Single octet identifying the algorithm. The following values are defined by this document:

0: Shortest Path First (SPF) algorithm based on link metric. This is the standard shortest path algorithm as computed by the OSPF protocol. Consistent with the deployed practice for link-state protocols, Algorithm 0 permits any node to overwrite the SPF path with a different path based on its local policy. If the SR-Algorithm TLV is advertised, Algorithm 0 MUST be included.

1: Strict Shortest Path First (SPF) algorithm based on link metric. The algorithm is identical to Algorithm 0 but Algorithm 1 requires that all nodes along the path will honor the SPF routing decision. Local policy at the node claiming support for Algorithm 1 MUST NOT alter the SPF paths computed by Algorithm 1.

When multiple SR-Algorithm TLVs are received from a given router, the receiver MUST use the first occurrence of the TLV in the Router Information LSA. If the SR-Algorithm TLV appears in multiple Router Information LSAs that have different flooding scopes, the SR-Algorithm TLV in the Router Information LSA with the area-scoped flooding scope MUST be used. If the SR-Algorithm TLV appears in

multiple Router Information LSAs that have the same flooding scope, the SR-Algorithm TLV in the Router Information (RI) LSA with the numerically smallest Instance ID MUST be used and subsequent instances of the SR-Algorithm TLV MUST be ignored.

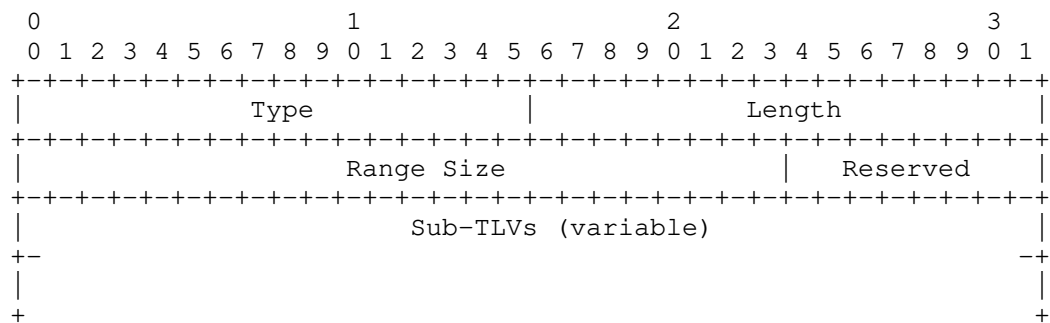
The RI LSA can be advertised at any of the defined opaque flooding scopes (link, area, or Autonomous System (AS)). For the purpose of SR-Algorithm TLV advertisement, area-scoped flooding is REQUIRED.

3.2. SID/Label Range TLV

Prefix SIDs MAY be advertised in a form of an index as described in Section 5. Such index defines the offset in the SID/Label space advertised by the router. The SID/Label Range TLV is used to advertise such SID/Label space.

The SID/Label Range TLV is a top-level TLV of the Router Information Opaque LSA (defined in [RFC7770]).

The SID/Label Range TLV MAY appear multiple times and has the following format:



where:

Type: 9

Length: Variable, in octets, dependent on Sub-TLVs.

Range Size: 3-octet SID/label range size (i.e., the number of SIDs or labels in the range including the first SID/label). It MUST be greater than 0.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

Initially, the only supported Sub-TLV is the SID/Label Sub-TLV as defined in Section 2.1. The SID/Label Sub-TLV MUST be included in the SID/Label Range TLV. The SID/Label advertised in the SID/Label Sub-TLV represents the first SID/Label in the advertised range.

Only a single SID/Label Sub-TLV MAY be advertised in SID/Label Range TLV. If more than one SID/Label Sub-TLVs are present, the SID/Label Range TLV MUST be ignored.

Multiple occurrences of the SID/Label Range TLV MAY be advertised, in order to advertise multiple ranges. In such case:

- o The originating router MUST encode each range into a different SID/Label Range TLV.
- o The originating router decides the order in which the set of SID/Label Range TLVs are advertised inside the Router Information Opaque LSA. The originating router MUST ensure the order is the same after a graceful restart (using checkpointing, non-volatile storage, or any other mechanism) in order to assure the SID/label range and SID index correspondence is preserved across graceful restarts.
- o The receiving router MUST adhere to the order in which the ranges are advertised when calculating a SID/label from a SID index.
- o The originating router MUST NOT advertise overlapping ranges.
- o When a router receives multiple overlapping ranges, it MUST conform to the procedures defined in [I-D.ietf-spring-segment-routing-mpls].

The following example illustrates the advertisement of multiple ranges:

The originating router advertises the following ranges:

```
Range 1: Range Size: 100    SID/Label Sub-TLV: 100
Range 1: Range Size: 100    SID/Label Sub-TLV: 1000
Range 1: Range Size: 100    SID/Label Sub-TLV: 500
```

The receiving routers concatenate the ranges and build the Segment Routing Global Block (SRGB) as follows:

```
SRGB = [100, 199]
       [1000, 1099]
       [500, 599]
```

The indexes span multiple ranges:

```
index=0 means label 100
...
index 99 means label 199
index 100 means label 1000
index 199 means label 1099
...
index 200 means label 500
...
```

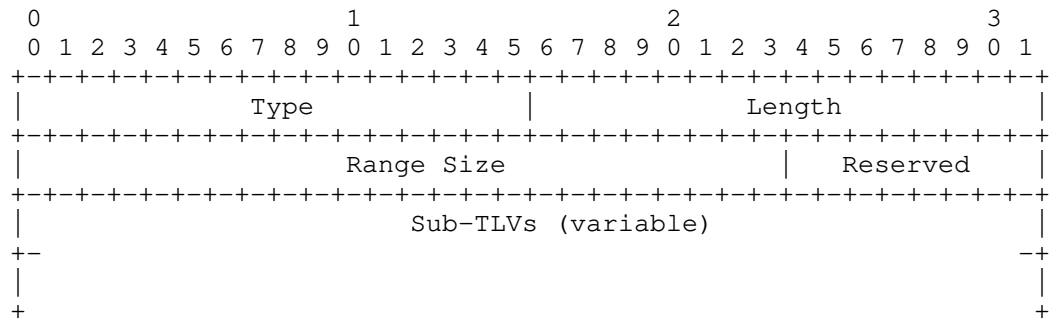
The RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). For the purpose of SID/Label Range TLV advertisement, area-scoped flooding is REQUIRED.

3.3. SR Local Block TLV

The SR Local Block TLV (SRLB TLV) contains the range of labels the node has reserved for local SIDs. SIDs from the SRLB MAY be used for Adjacency-SIDs, but also by components other than the OSPF protocol. As an example, an application or a controller can instruct the router to allocate a specific local SID. Some controllers or applications can use the control plane to discover the available set of local SIDs on a particular router. In such cases, the SRLB is advertised in the control plane. The requirement to advertise the SRLB is further described in [I-D.ietf-spring-segment-routing-mpls]. The SRLB TLV is used to advertise the SRLB.

The SRLB TLV is a top-level TLV of the Router Information Opaque LSA (defined in [RFC7770]).

The SRLB TLV MAY appear multiple times in the Router Information Opaque LSA and has the following format:



where:

Type: 14

Length: Variable, in octets, dependent on Sub-TLVs.

Range Size: 3-octet SID/label range size (i.e., the number of SIDs or labels in the range including the first SID/label). It MUST be greater than 0.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

Initially, the only supported Sub-TLV is the SID/Label Sub-TLV as defined in Section 2.1. The SID/Label Sub-TLV MUST be included in the SRLB TLV. The SID/Label advertised in the SID/Label Sub-TLV represents the first SID/Label in the advertised range.

Only a single SID/Label Sub-TLV MAY be advertised in the SRLB TLV. If more than one SID/Label Sub-TLVs are present, the SRLB TLV MUST be ignored.

The originating router MUST NOT advertise overlapping ranges.

Each time a SID from the SRLB is allocated, it SHOULD also be reported to all components (e.g., controller or applications) in order for these components to have an up-to-date view of the current SRLB allocation. This is required to avoid collisions between allocation instructions.

Within the context of OSPF, the reporting of local SIDs is done through OSPF Sub-TLVs such as the Adjacency-SID (Section 6). However, the reporting of allocated local SIDs can also be done through other means and protocols which are outside the scope of this document.

A router advertising the SRLB TLV MAY also have other label ranges, outside of the SRLB, used for its local allocation purposes which are not advertised in the SRLB TLV. For example, it is possible that an Adjacency-SID is allocated using a local label that is not part of the SRLB.

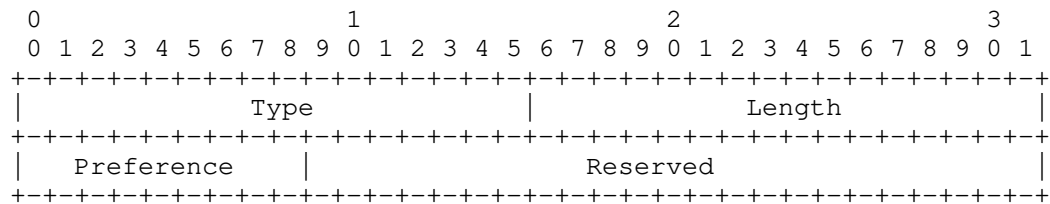
The RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). For the purpose of SRLB TLV advertisement, area-scoped flooding is REQUIRED.

3.4. SRMS Preference TLV

The Segment Routing Mapping Server Preference TLV (SRMS Preference TLV) is used to advertise a preference associated with the node that acts as an SR Mapping Server. The role of an SRMS is described in [I-D.ietf-spring-segment-routing-ldp-interop]. SRMS preference is defined in [I-D.ietf-spring-segment-routing-ldp-interop].

The SRMS Preference TLV is a top-level TLV of the Router Information Opaque LSA (defined in [RFC7770]).

The SRMS Preference TLV MAY only be advertised once in the Router Information Opaque LSA and has the following format:



where:

Type: 15

Length: 4 octets

Preference: 1 octet. SRMS preference value from 0 to 255.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

When multiple SRMS Preference TLVs are received from a given router, the receiver MUST use the first occurrence of the TLV in the Router Information LSA. If the SRMS Preference TLV appears in multiple Router Information LSAs that have different flooding scopes, the SRMS Preference TLV in the Router Information LSA with the narrowest

flooding scope MUST be used. If the SRMS Preference TLV appears in multiple Router Information LSAs that have the same flooding scope, the SRMS Preference TLV in the Router Information LSA with the numerically smallest Instance ID MUST be used and subsequent instances of the SRMS Preference TLV MUST be ignored.

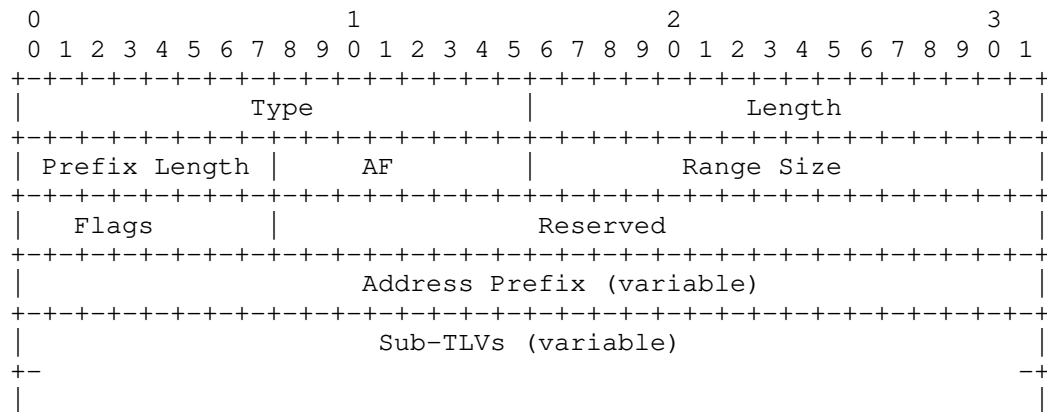
The RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). For the purpose of the SRMS Preference TLV advertisement, AS-scoped flooding SHOULD be used. This is because SRMS servers can be located in a different area than consumers of the SRMS advertisements. If the SRMS advertisements from the SRMS server are only used inside the SRMS server's area, area-scoped flooding MAY be used.

4. OSPF Extended Prefix Range TLV

In some cases it is useful to advertise attributes for a range of prefixes. The Segment Routing Mapping Server, which is described in [I-D.ietf-spring-segment-routing-ldp-interop], is an example where we need a single advertisement to advertise SIDs for multiple prefixes from a contiguous address range.

The OSPF Extended Prefix Range TLV, which is a top level TLV of the Extended Prefix LSA described in [RFC7684] is defined for this purpose.

Multiple OSPF Extended Prefix Range TLVs MAY be advertised in each OSPF Extended Prefix Opaque LSA, but all prefix ranges included in a single OSPF Extended Prefix Opaque LSA MUST have the same flooding scope. The OSPF Extended Prefix Range TLV has the following format:



where:

Type: 2

Length: Variable, in octets, dependent on Sub-TLVs.

Prefix length: Length of prefix in bits.

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.

Range size: Represents the number of prefixes that are covered by the advertisement. The Range Size MUST NOT exceed the number of prefixes that could be satisfied by the prefix length without including the IPv4 multicast address range (224.0.0.0/3).

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

0	1	2	3	4	5	6	7
IA							

where:

IA-Flag: Inter-Area flag. If set, advertisement is of inter-area type. An ABR that is advertising the OSPF Extended Prefix Range TLV between areas MUST set this bit.

This bit is used to prevent redundant flooding of Prefix Range TLVs between areas as follows:

An ABR only propagates an inter-area Prefix Range advertisement from the backbone area to connected non-backbone areas if the advertisement is considered to be the best one. The following rules are used to select the best range from the set of advertisements for the same Prefix Range:

An ABR always prefers intra-area Prefix Range advertisements over inter-area advertisements.

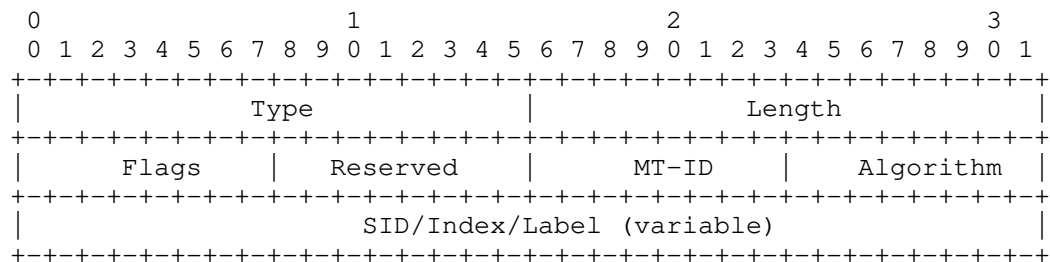
An ABR does not consider inter-area Prefix Range advertisements coming from non-backbone areas.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

Address Prefix: 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. Prefix encoding for other address families is beyond the scope of this specification.

5. Prefix SID Sub-TLV

The Prefix SID Sub-TLV is a Sub-TLV of the OSPF Extended Prefix TLV described in [RFC7684] and the OSPF Extended Prefix Range TLV described in Section 4. It MAY appear more than once in the parent TLV and has the following format:

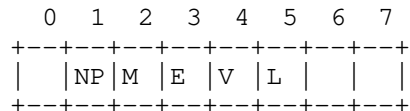


where:

Type: 2

Length: 7 or 8 octets, dependent on the V-flag

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



where:

NP-Flag: No-PHP flag. If set, then the penultimate hop MUST NOT pop the Prefix-SID before delivering packets to the node that advertised the Prefix-SID.

M-Flag: Mapping Server Flag. If set, the SID was advertised by a Segment Routing Mapping Server as described in [I-D.ietf-spring-segment-routing-ldp-interop].

E-Flag: Explicit-Null Flag. If set, any upstream neighbor of the Prefix-SID originator MUST replace the Prefix-SID with the Explicit-NULL label (0 for IPv4) before forwarding the packet.

V-Flag: Value/Index Flag. If set, then the Prefix-SID carries an absolute value. If not set, then the Prefix-SID carries an index.

L-Flag: Local/Global Flag. If set, then the value/index carried by the Prefix-SID has local significance. If not set, then the value/index carried by this Sub-TLV has global significance.

Other bits: Reserved. These MUST be zero when sent and are ignored when received.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

MT-ID: Multi-Topology ID (as defined in [RFC4915]).

Algorithm: Single octet identifying the algorithm the Prefix-SID is associated with as defined in Section 3.1.

A router receiving a Prefix-SID from a remote node and with an algorithm value that such remote node has not advertised in the SR-Algorithm Sub-TLV (Section 3.1) MUST ignore the Prefix-SID Sub-TLV.

SID/Index/Label: According to the V and L flags, it contains:

V-flag is set to 0 and L-flag is set to 0: The SID/Index/Label field is a 4 octet index defining the offset in the SID/Label space advertised by this router

V-flag is set to 1 and L-flag is set to 1: The SID/Index/Label field is a 3 octet local label where the 20 rightmost bits are used for encoding the label value.

All other combinations of V-flag and L-flag are invalid and any SID advertisement received with an invalid setting for V and L flags MUST be ignored.

If an OSPF router advertises multiple Prefix-SIDs for the same prefix, topology and algorithm, all of them MUST be ignored.

When calculating the outgoing label for the prefix, the router MUST take into account, as described below, the E, NP and M flags

advertised by the next-hop router if that router advertised the SID for the prefix. This MUST be done regardless of whether the next-hop router contributes to the best path to the prefix.

The NP-Flag (No-PHP) MUST be set and the E-flag MUST be clear for Prefix-SIDs allocated to inter-area prefixes that are originated by the ABR based on intra-area or inter-area reachability between areas, unless the advertised prefix is directly attached to the ABR.

The NP-Flag (No-PHP) MUST be set and the E-flag MUST be clear for Prefix-SIDs allocated to redistributed prefixes, unless the redistributed prefix is directly attached to the ASBR.

If the NP-Flag is not set, then any upstream neighbor of the Prefix-SID originator MUST pop the Prefix-SID. This is equivalent to the penultimate hop popping mechanism used in the MPLS dataplane. If the NP-flag is not set, then the received E-flag is ignored.

If the NP-flag is set then:

If the E-flag is not set, then any upstream neighbor of the Prefix-SID originator MUST keep the Prefix-SID on top of the stack. This is useful when the originator of the Prefix-SID need to stitch the incoming packet into a continuing MPLS LSP to the final destination. This could occur at an Area Border Router (prefix propagation from one area to another) or at an AS Boundary Router (prefix propagation from one domain to another).

If the E-flag is set, then any upstream neighbor of the Prefix-SID originator MUST replace the Prefix-SID with an Explicit-NULL label. This is useful, e.g., when the originator of the Prefix-SID is the final destination for the related prefix and the originator wishes to receive the packet with the original EXP bits.

When the M-Flag is set, the NP-flag and the E-flag MUST be ignored at reception.

As the Mapping Server does not specify the originator of a prefix advertisement, it is not possible to determine PHP behavior solely based on the Mapping Server advertisement. However, PHP behavior SHOULD be done in following cases:

The Prefix is intra-area type and the downstream neighbor is the originator of the prefix.

The Prefix is inter-area type and downstream neighbor is an ABR, which is advertising prefix reachability and is also generating

the Extended Prefix TLV with the A-flag set for this prefix as described in section 2.1 of [RFC7684].

The Prefix is external type and downstream neighbor is an ASBR, which is advertising prefix reachability and is also generating the Extended Prefix TLV with the A-flag set for this prefix as described in section 2.1 of [RFC7684].

When a Prefix-SID is advertised in an Extended Prefix Range TLV, then the value advertised in the Prefix SID Sub-TLV is interpreted as a starting SID/Label value.

Example 1: If the following router addresses (loopback addresses) need to be mapped into the corresponding Prefix SID indexes:

```
Router-A: 192.0.2.1/32, Prefix-SID: Index 1
Router-B: 192.0.2.2/32, Prefix-SID: Index 2
Router-C: 192.0.2.3/32, Prefix-SID: Index 3
Router-D: 192.0.2.4/32, Prefix-SID: Index 4
```

then the Prefix field in the Extended Prefix Range TLV would be set to 192.0.2.1, Prefix Length would be set to 32, Range Size would be set to 4, and the Index value in the Prefix-SID Sub-TLV would be set to 1.

Example 2: If the following prefixes need to be mapped into the corresponding Prefix-SID indexes:

```
192.0.2.0/30, Prefix-SID: Index 51
192.0.2.4/30, Prefix-SID: Index 52
192.0.2.8/30, Prefix-SID: Index 53
192.0.2.12/30, Prefix-SID: Index 54
192.0.2.16/30, Prefix-SID: Index 55
192.0.2.20/30, Prefix-SID: Index 56
192.0.2.24/30, Prefix-SID: Index 57
```

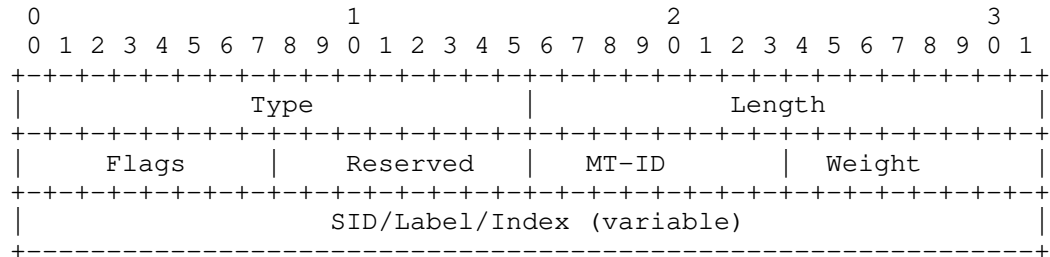
then the Prefix field in the Extended Prefix Range TLV would be set to 192.0.2.0, Prefix Length would be set to 30, Range Size would be 7, and the Index value in the Prefix-SID Sub-TLV would be set to 51.

6. Adjacency Segment Identifier (Adj-SID)

An Adjacency Segment Identifier (Adj-SID) represents a router adjacency in Segment Routing.

6.1. Adj-SID Sub-TLV

Adj-SID is an optional Sub-TLV of the Extended Link TLV defined in [RFC7684]. It MAY appear multiple times in the Extended Link TLV. The Adj-SID Sub-TLV has the following format:

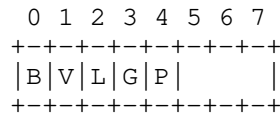


where:

Type: 2

Length: 7 or 8 octets, dependent on the V flag.

Flags: Single octet field containing the following flags:



where:

B-Flag: Backup Flag. If set, the Adj-SID refers to an adjacency that is eligible for protection (e.g., using IPFRR or MPLS-FRR) as described in section 3.5 of [I-D.ietf-spring-segment-routing].

The V-Flag: Value/Index Flag. If set, then the Adj-SID carries an absolute value. If not set, then the Adj-SID carries an index.

The L-Flag: Local/Global Flag. If set, then the value/index carried by the Adj-SID has local significance. If not set, then the value/index carried by this Sub-TLV has global significance.

The G-Flag: Group Flag. When set, the G-Flag indicates that the Adj-SID refers to a group of adjacencies (and therefore MAY be assigned to other adjacencies as well).

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

Other bits: Reserved. These MUST be zero when sent and are ignored when received.

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

MT-ID: Multi-Topology ID (as defined in [RFC4915]).

Weight: Weight used for load-balancing purposes. The use of the weight is defined in [I-D.ietf-spring-segment-routing].

SID/Index/Label: as described in Section 5.

An SR capable router MAY allocate an Adj-SID for each of its adjacencies and set the B-Flag when the adjacency is eligible for protection by an FRR mechanism (IP or MPLS) as described in section 3.5 of [I-D.ietf-spring-segment-routing].

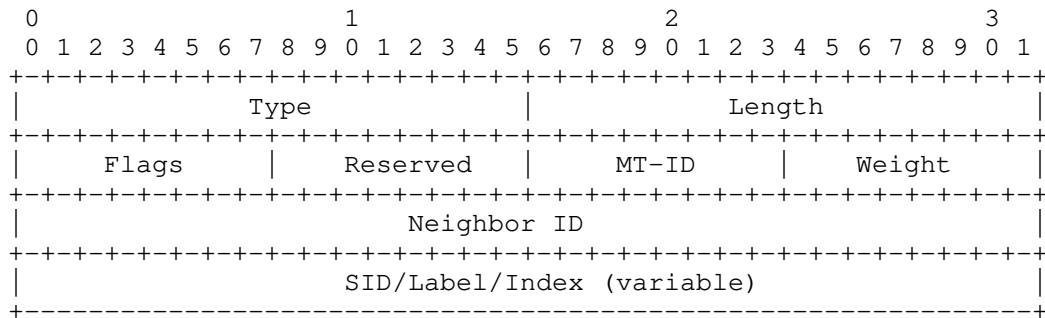
An SR capable router MAY allocate more than one Adj-SID to an adjacency

An SR capable router MAY allocate the same Adj-SID to different adjacencies

When the P-flag is not set, the Adj-SID MAY be persistent. When the P-flag is set, the Adj-SID MUST be persistent.

6.2. LAN Adj-SID Sub-TLV

LAN Adj-SID is an optional Sub-TLV of the Extended Link TLV defined in [RFC7684]. It MAY appear multiple times in the Extended-Link TLV. It is used to advertise a SID/Label for an adjacency to a non-DR router on a broadcast, NBMA, or hybrid [RFC6845] network.



where:

Type: 3

Length: 11 or 12 octets, dependent on V-flag.

Flags: same as in Section 6.1

Reserved: SHOULD be set to 0 on transmission and MUST be ignored on reception.

MT-ID: Multi-Topology ID (as defined in [RFC4915]).

Weight: Weight used for load-balancing purposes. The use of the weight is defined in [I-D.ietf-spring-segment-routing].

Neighbor ID: The Router ID of the neighbor for which the LAN-Adj-SID is advertised.

SID/Index/Label: as described in Section 5.

When the P-flag is not set, the Adj-SID MAY be persistent. When the P-flag is set, the Adj-SID MUST be persistent.

7. Elements of Procedure

7.1. Intra-area Segment routing in OSPFv2

An OSPFv2 router that supports segment routing MAY advertise Prefix-SIDs for any prefix to which it is advertising reachability (e.g., a loopback IP address as described in Section 5).

A Prefix-SID can also be advertised by the SR Mapping Servers (as described in [I-D.ietf-spring-segment-routing-ldp-interop]). A Mapping Server advertises Prefix-SIDs for remote prefixes that exist in the OSPFv2 routing domain. Multiple Mapping Servers can advertise

Prefix-SIDs for the same prefix, in which case the same Prefix-SID MUST be advertised by all of them. The flooding scope of the OSPF Extended Prefix Opaque LSA that is generated by the SR Mapping Server could be either area-scoped or AS-scoped and is determined based on the configuration of the SR Mapping Server.

An SR Mapping Server MUST use the OSPF Extended Prefix Range TLV when advertising SIDs for prefixes. Prefixes of different route-types can be combined in a single OSPF Extended Prefix Range TLV advertised by an SR Mapping Server. Because the OSPF Extended Prefix Range TLV doesn't include a Route-Type field, as in the OSPF Extended Prefix TLV, it is possible to include adjacent prefixes from different Route-Types in the OSPF Extended Prefix Range TLV.

Area-scoped OSPF Extended Prefix Range TLVs are propagated between areas. Similar to propagation of prefixes between areas, an ABR only propagates the OSPF Extended Prefix Range TLV that it considers to be the best from the set it received. The rules used to pick the best OSPF Extended Prefix Range TLV are described in Section 4.

When propagating an OSPF Extended Prefix Range TLV between areas, ABRs MUST set the IA-Flag, that is used to prevent redundant flooding of the OSPF Extended Prefix Range TLV between areas as described in Section 4.

7.2. Inter-area Segment routing in OSPFv2

In order to support SR in a multi-area environment, OSPFv2 MUST propagate Prefix-SID information between areas. The following procedure is used to propagate Prefix SIDs between areas.

When an OSPF ABR advertises a Type-3 Summary LSA from an intra-area prefix to all its connected areas, it will also originate an Extended Prefix Opaque LSA, as described in [RFC7684]. The flooding scope of the Extended Prefix Opaque LSA type will be set to area-local scope. The route-type in the OSPF Extended Prefix TLV is set to inter-area. The Prefix-SID Sub-TLV will be included in this LSA and the Prefix-SID value will be set as follows:

The ABR will look at its best path to the prefix in the source area and find the advertising router associated with the best path to that prefix.

The ABR will then determine if such router advertised a Prefix-SID for the prefix and use it when advertising the Prefix-SID to other connected areas.

If no Prefix-SID was advertised for the prefix in the source area by the router that contributes to the best path to the prefix, the originating ABR will use the Prefix-SID advertised by any other router when propagating the Prefix-SID for the prefix to other areas.

When an OSPF ABR advertises Type-3 Summary LSAs from an inter-area route to all its connected areas, it will also originate an Extended Prefix Opaque LSA, as described in [RFC7684]. The flooding scope of the Extended Prefix Opaque LSA type will be set to area-local scope. The route-type in OSPF Extended Prefix TLV is set to inter-area. The Prefix-SID Sub-TLV will be included in this LSA and the Prefix-SID will be set as follows:

The ABR will look at its best path to the prefix in the backbone area and find the advertising router associated with the best path to that prefix.

The ABR will then determine if such router advertised a Prefix-SID for the prefix and use it when advertising the Prefix-SID to other connected areas.

If no Prefix-SID was advertised for the prefix in the backbone area by the ABR that contributes to the best path to the prefix, the originating ABR will use the Prefix-SID advertised by any other router when propagating the Prefix-SID for the prefix to other areas.

7.3. Segment Routing for External Prefixes

Type-5 LSAs are flooded domain wide. When an ASBR, which supports SR, generates Type-5 LSAs, it SHOULD also originate Extended Prefix Opaque LSAs, as described in [RFC7684]. The flooding scope of the Extended Prefix Opaque LSA type is set to AS-wide scope. The route-type in the OSPF Extended Prefix TLV is set to external. The Prefix-SID Sub-TLV is included in this LSA and the Prefix-SID value will be set to the SID that has been reserved for that prefix.

When an NSSA [RFC3101] ABR translates Type-7 LSAs into Type-5 LSAs, it SHOULD also advertise the Prefix-SID for the prefix. The NSSA ABR determines its best path to the prefix advertised in the translated Type-7 LSA and finds the advertising router associated with that path. If the advertising router has advertised a Prefix-SID for the prefix, then the NSSA ABR uses it when advertising the Prefix-SID for the Type-5 prefix. Otherwise, the Prefix-SID advertised by any other router will be used.

7.4. Advertisement of Adj-SID

The Adjacency Segment Routing Identifier (Adj-SID) is advertised using the Adj-SID Sub-TLV as described in Section 6.

7.4.1. Advertisement of Adj-SID on Point-to-Point Links

An Adj-SID MAY be advertised for any adjacency on a P2P link that is in neighbor state 2-Way or higher. If the adjacency on a P2P link transitions from the FULL state, then the Adj-SID for that adjacency MAY be removed from the area. If the adjacency transitions to a state lower than 2-Way, then the Adj-SID advertisement MUST be withdrawn from the area.

7.4.2. Adjacency SID on Broadcast or NBMA Interfaces

Broadcast, NBMA, or hybrid [RFC6845] networks in OSPF are represented by a star topology where the Designated Router (DR) is the central point to which all other routers on the broadcast, NBMA, or hybrid network connect. As a result, routers on the broadcast, NBMA, or hybrid network advertise only their adjacency to the DR. Routers that do not act as DR do not form or advertise adjacencies with each other. They do, however, maintain 2-Way adjacency state with each other and are directly reachable.

When Segment Routing is used, each router on the broadcast, NBMA, or hybrid network MAY advertise the Adj-SID for its adjacency to the DR using the Adj-SID Sub-TLV as described in Section 6.1.

SR capable routers MAY also advertise a LAN-Adj-SID for other neighbors (e.g., BDR, DR-OTHER) on the broadcast, NBMA, or hybrid network using the LAN-ADJ-SID Sub-TLV as described in Section 6.2.

8. IANA Considerations

This specification updates several existing OSPF registries.

8.1. OSPF Router Information (RI) TLVs Registry

- o 8 (IANA Preallocated) - SR-Algorithm TLV
- o 9 (IANA Preallocated) - SID/Label Range TLV
- o 14 - SR Local Block TLV
- o 15 - SRMS Preference TLV

8.2. OSPFv2 Extended Prefix Opaque LSA TLVs Registry

Following values are allocated:

- o 2 - OSPF Extended Prefix Range TLV

8.3. OSPFv2 Extended Prefix TLV Sub-TLVs Registry

Following values are allocated:

- o 1 - SID/Label Sub-TLV
- o 2 - Prefix SID Sub-TLV

8.4. OSPFv2 Extended Link TLV Sub-TLVs Registry

Following initial values are allocated:

- o 1 - SID/Label Sub-TLV
- o 2 - Adj-SID Sub-TLV
- o 3 - LAN Adj-SID/Label Sub-TLV

8.5. IGP Algorithm Type Registry

IANA is requested to set up a registry called "IGP Algorithm Type" under a new category of "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 registry are:

0: Shortest Path First (SPF) algorithm based on link metric. This is the standard shortest path algorithm as computed by the IGP protocol. Consistent with the deployed practice for link-state protocols, Algorithm 0 permits any node to overwrite the SPF path with a different path based on its local policy.

1: Strict Shortest Path First (SPF) algorithm based on link metric. The algorithm is identical to Algorithm 0 but Algorithm 1 requires that all nodes along the path will honor the SPF routing decision. Local policy at the node claiming support for Algorithm 1 MUST NOT alter the SPF paths computed by Algorithm 1.

9. Implementation Status

An implementation survey with seven questions related to the implementer's support of OSPFv2 Segment Routing was sent to the OSPF WG list and several known implementers. This section contains responses from three implementers who completed the survey. No external means were used to verify the accuracy of the information submitted by the respondents. The respondents are considered experts on the products they reported on. Additionally, responses were omitted from implementers who indicated that they have not implemented the function yet.

This section will be removed before publication as an RFC.

Responses from Nokia (former Alcatel-Lucent):

Link to a web page describing the implementation:

https://infoproducts.alcatel-lucent.com/cgi-bin/dbaccessfilename.cgi/3HE10799AAAATQZZA01_V1_7450%20ESS%207750%20SR%20and%207950%20XRS%20Unicast%20Routing%20Protocols%20Guide%20R14.0.R1.pdf

The implementation's level of maturity: Production.

Coverage: We have implemented all sections and have support for the latest draft.

Licensing: Part of the software package that needs to be purchased.

Implementation experience: Great spec. We also performed inter-operability testing with Cisco's OSPF Segment Routing implementation.

Contact information: wim.henderickx@nokia.com

Responses from Cisco Systems:

Link to a web page describing the implementation:

<http://www.segment-routing.net/home/tutorial>

The implementation's level of maturity: Production.

Coverage: All sections have been implemented according to the latest draft.

Licensing: Part of a commercial software package.

Implementation experience: Many aspects of the draft are result of the actual implementation experience, as the draft evolved from its

initial version to the current one. Interoperability testing with Alcatel-Lucent was performed, which confirmed the draft's ability to serve as a reference for the implementors.

Contact information: ppsenak@cisco.com

Responses from Juniper:

The implementation's name and/or a link to a web page describing the implementation:

Feature name is OSPF SPRING

The implementation's level of maturity: To be released in 16.2 (second half of 2016)

Coverage: All sections implemented except Sections 4, and 6.

Licensing: JUNOS Licensing needed.

Implementation experience: NA

Contact information: shraddha@juniper.net

10. Security Considerations

With the OSPFv2 segment routing extensions defined herein, OSPFv2 will now program the MPLS data plane [RFC3031] in addition to the IP data plane. Previously, LDP [RFC5036] or another label distribution mechanism was required to advertise MPLS labels and program the MPLS data plane.

In general, the same types of attacks that can be carried out on the IP control plane can be carried out on the MPLS control plane resulting in traffic being misrouted in the respective data planes. However, the latter can be more difficult to detect and isolate.

Existing security extensions as described in [RFC2328] and [RFC7684] apply to these segment routing extensions. 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.

Implementations MUST assure that malformed TLV and Sub-TLV defined in this document are detected and do not provide a vulnerability for attackers to crash the OSPFv2 router or routing process. Reception of malformed TLV or Sub-TLV SHOULD be counted and/or logged for

further analysis. Logging of malformed TLVs and Sub-TLVs SHOULD be rate-limited to prevent a Denial of Service (DoS) attack (distributed or otherwise) from overloading the OSPF control plane.

11. Contributors

The following people gave a substantial contribution to the content of this document: Acee Lindem, Ahmed Bashandy, Martin Horneffer, Bruno Decraene, Stephane Litkowski, Igor Milojevic, Rob Shakir and Saku Ytti.

12. Acknowledgements

We would like to thank Anton Smirnov for his contribution.

Thanks to Acee Lindem for the detail review of the draft, corrections, as well as discussion about details of the encoding.

13. References

13.1. Normative 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.
- [I-D.ietf-spring-segment-routing-ldp-interop]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., and S. Litkowski, "Segment Routing interworking with LDP", draft-ietf-spring-segment-routing-ldp-interop-15 (work in progress), September 2018.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-15 (work in progress), October 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>>.

- [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>>.
- [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>>.
- [RFC6845] Sheth, N., Wang, L., and J. Zhang, "OSPF Hybrid Broadcast and Point-to-Multipoint Interface Type", RFC 6845, DOI 10.17487/RFC6845, January 2013, <<https://www.rfc-editor.org/info/rfc6845>>.
- [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>>.
- [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>>.
- [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>>.

13.2. Informative References

- [I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P. and S. Previdi, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-18 (work in progress), November 2018.
- [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>>.

[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>>.

Authors' Addresses

Peter Psenak (editor)
Cisco Systems, Inc.
Apollo Business Center
Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: stefano@previdi.net

Clarence Filsfils
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

Rob Shakir
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: robjs@google.com

Wim Henderickx
Nokia
Copernicuslaan 50
Antwerp 2018
BE

Email: wim.henderickx@nokia.com

Jeff Tantsura
Apstra, Inc.

Email: jefftant.ietf@gmail.com

Ospf Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 29, 2015

Q. Liang
J. You
N. Wu
Huawei
P. Fan
China Mobile
K. Patel
A. Lindem
Cisco Systems
May 28, 2015

OSPF Extensions for Flow Specification
draft-liang-ospf-flowspec-extensions-05

Abstract

Dissemination of the Traffic flow information was first introduced in the BGP protocol [RFC5575]. FlowSpec routes are used to distribute traffic filtering rules that are used to filter Denial-of-Service (DoS) attacks. For the networks that only deploy an IGP (Interior Gateway Protocol) (e.g., OSPF), it is required that the IGP is extended to distribute Flow Specification or FlowSpec routes.

This document discusses the use cases for distributing flow specification (FlowSpec) routes using OSPF. Furthermore, this document defines a OSPF FlowSpec Opaque Link State Advertisement (LSA) encoding format that can be used to distribute FlowSpec routes, its validation procedures for imposing the filtering information on the routers, and a capability to indicate the support of FlowSpec functionality.

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 <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 November 29, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. Use Cases for OSPF based FlowSpec Distribution	4
3.1. OSPF Campus Network	4
3.2. BGP/MPLS VPN	4
3.2.1. Traffic Analyzer Deployed in Provider Network	5
3.2.2. Traffic Analyzer Deployed in Customer Network	6
3.2.3. Policy Configuration	6
4. OSPF Extensions for FlowSpec Rules	7
4.1. FlowSpec LSA	7
4.1.1. OSPFv2 FlowSpec Opaque LSA	7
4.1.2. OSPFv3 FlowSpec LSA	9
4.2. OSPF FlowSpec Filters TLV	10
4.2.1. Order of Traffic Filtering Rules	11
4.2.2. Validation Procedure	12
4.3. OSPF FlowSpec Action TLV	12
4.3.1. Traffic-rate	13
4.3.2. Traffic-action	13
4.3.3. Traffic-marking	13
4.3.4. Redirect-to-IP	14
4.4. Capability Advertisement	15
5. Redistribution of FlowSpec Routes	15
6. IANA Considerations	15
7. Security considerations	16

8. Acknowledgement	16
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Authors' Addresses	17

1. Introduction

[RFC5575] defines Border Gateway Protocol protocol extensions that can be used to distribute traffic flow specifications. One application of this encoding format is to automate inter-domain coordination of traffic filtering, such as what is required in order to mitigate (distributed) denial-of-service attacks. [RFC5575] allows flow specifications received from an external autonomous system to be forwarded to a given BGP peer. However, in order to block the attack traffic more effectively, it is better to distribute the BGP FlowSpec routes to the customer network, which is much closer to the attacker.

For the networks deploying only an IGP (e.g., OSPF), it is expected to extend the IGP (OSPF in this document) to distribute FlowSpec routes. This document discusses the use cases for distributing FlowSpec routes using OSPF. Furthermore, this document also defines a new OSPF FlowSpec Opaque Link State Advertisement (LSA) [RFC5250] encoding format that can be used to distribute FlowSpec routes to the edge routers in the customer network, its validation procedures for imposing the filtering information on the routers, and a capability to indicate the support of Flowspec functionality.

The semantic content of the FlowSpec extensions defined in this document are identical to the corresponding extensions to BGP ([RFC5575] and [I-D.ietf-idr-flow-spec-v6]). In order to avoid repetition, this document only concentrates on those parts of specification where OSPF is different from BGP. The OSPF flowspec extensions defined in this document can be used to mitigate the impacts of DoS attacks.

2. Terminology

This section contains definitions for terms used frequently throughout this document. However, many additional definitions can be found in [RFC5250] and [RFC5575].

Flow Specification (FlowSpec): A flow specification is an n-tuple consisting of several matching criteria that can be applied to IP traffic, including filters and actions. Each FlowSpec consists of a set of filters and a set of actions.

3. Use Cases for OSPF based FlowSpec Distribution

For the networks deploying only an IGP (e.g., OSPF), it is expected to extend the IGP (OSPF in this document) to distribute FlowSpec routes, because when the FlowSpec routes are installed in the customer network, they are closer to the attacker than when they are installed in the provider network. Consequently, the attack traffic could be blocked or the suspicious traffic could be limited to a low rate as early as possible.

The following sub-sections discuss the use cases for OSPF based FlowSpec route distribution.

3.1. OSPF Campus Network

For networks not deploying BGP, for example, the campus network using OSPF, it is expected to extend OSPF to distribute FlowSpec routes as shown in Figure 3. In this kind of network, the traffic analyzer could be deployed with a router, then the FlowSpec routes from the traffic analyzer need to be distributed to the other routers in this domain using OSPF.

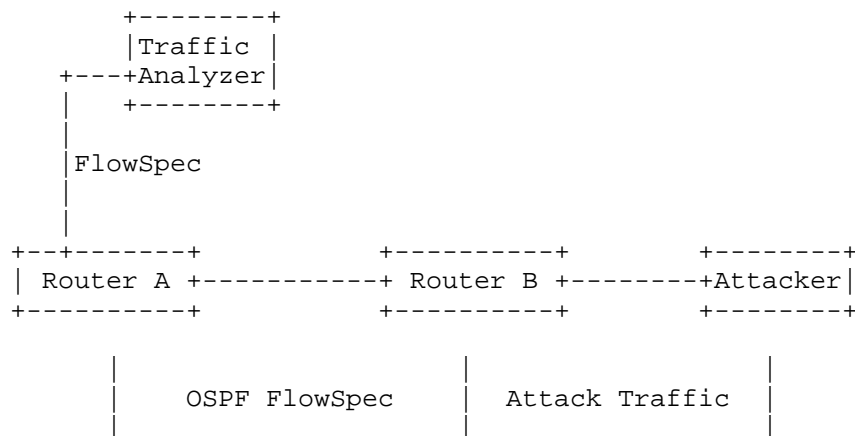


Figure 3: OSPF Campus Network

3.2. BGP/MPLS VPN

[RFC5575] defines a BGP NLRI encoding format to distribute traffic flow specifications in BGP deployed network. However, in the BGP/MPLS VPN scenario, the IGP (e.g., IS-IS, or OSPF) is used between the PE (Provider Edge) and CE (Customer Edge) in many deployments. In order to distribute the FlowSpec routes to the customer network, the IGP needs to support FlowSpec route distribution. The FlowSpec

routes are usually generated by the traffic analyzer or the traffic policy center in the network. Depending on the location of the traffic analyzer deployment, two different distribution scenarios are discussed below.

3.2.1. Traffic Analyzer Deployed in Provider Network

The traffic analyzer (also acting as the traffic policy center) could be deployed in the provider network as shown in Figure 1. If the traffic analyzer detects attack traffic from the customer network VPN1, it would generate the FlowSpec routes for preventing DoS attacks. FlowSpec routes with a Route Distinguisher (RD) in the Network Layer Reachability information (NRLI) corresponding to VPN1 are distributed from the traffic analyzer to the PE1 to which the traffic analyzer is attached. If the traffic analyzer is also a BGP speaker, it can distribute the FlowSpec routes using BGP [RFC5575]. Then the PE1 distributes the FlowSpec routes further to the PE2. Finally, the FlowSpec routes need to be distributed from PE2 to the CE2 using OSPF, i.e., to the customer network VPN1. As an attacker is more likely in the customer network, FlowSpec routes installed directly on CE2 could mitigate the impact of DoS attacks better.

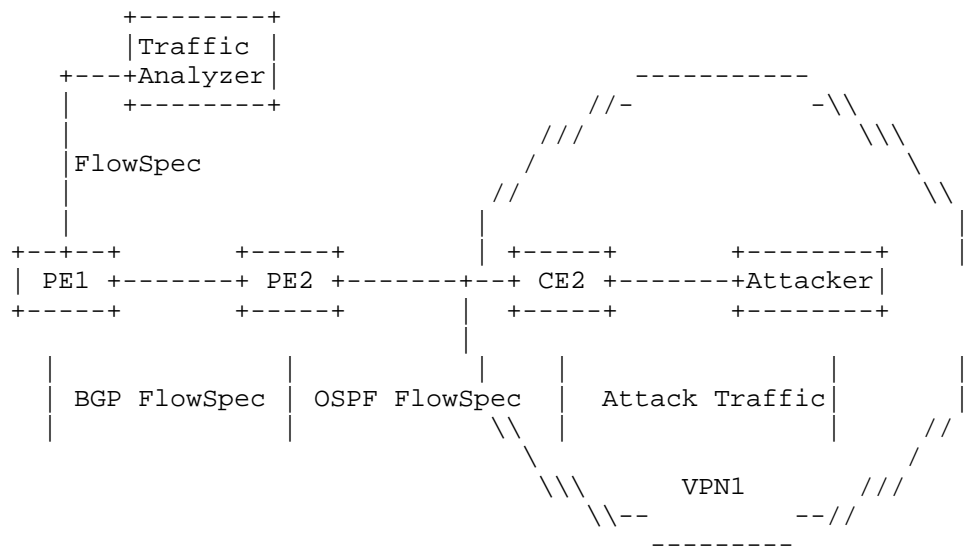


Figure 1: Traffic Analyzer deployed in Provider Network

3.2.2. Traffic Analyzer Deployed in Customer Network

The traffic analyzer (also acting as the traffic policy center) could be deployed in the customer network as shown in Figure 2. If the traffic analyzer detects attack traffic, it would generate FlowSpec routes to prevent associated DoS attacks. Then the FlowSpec routes would be distributed from the traffic analyzer to the CE1 using OSPF or another policy protocol (e.g., RESTful API over HTTP). Furthermore, the FlowSpec routes need to be distributed throughout the provider network via PE1/PE2 to CE2, i.e., to the remote customer network VPN1 Site1. If the FlowSpec routes installed on the CE2, it could block the attack traffic as close to the source of the attack as possible.

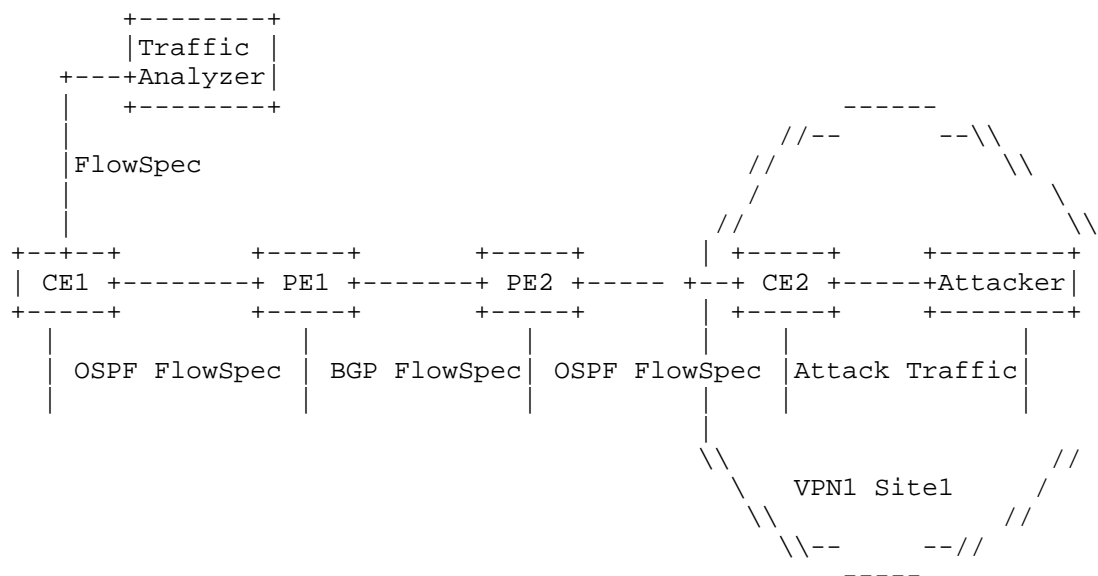


Figure 2: Traffic Analyzer deployed in Customer Network

3.2.3. Policy Configuration

The CE or PE could deploy local filtering policies to filter OSPF FlowSpec rules, for example, deploying a filtering policy to filter the incoming OSPF FlowSpec rules in order to prevent illegal or invalid FlowSpec rules from being applied.

The PE should configure FlowSpec importing policies to control importing action between the BGP IP/VPN FlowSpec RIB and the OSPF Instance FlowSpec RIB. Otherwise, the PE couldn't transform a BGP

IP/VPN FlowSpec rule to an OSPF FlowSpec rule or transform an OSPF FlowSpec rule to a BGP IP/VPN FlowSpec rule.

4. OSPF Extensions for FlowSpec Rules

4.1. FlowSpec LSA

4.1.1. OSPFv2 FlowSpec Opaque LSA

This document defines a new OSPFv2 flow specification Opaque Link State Advertisement (LSA) encoding format that can be used to distribute traffic flow specifications. This new OSPF FlowSpec Opaque LSA is extended based on [RFC5250].

The OSPFv2 FlowSpec Opaque LSA is defined below in Figure 4:

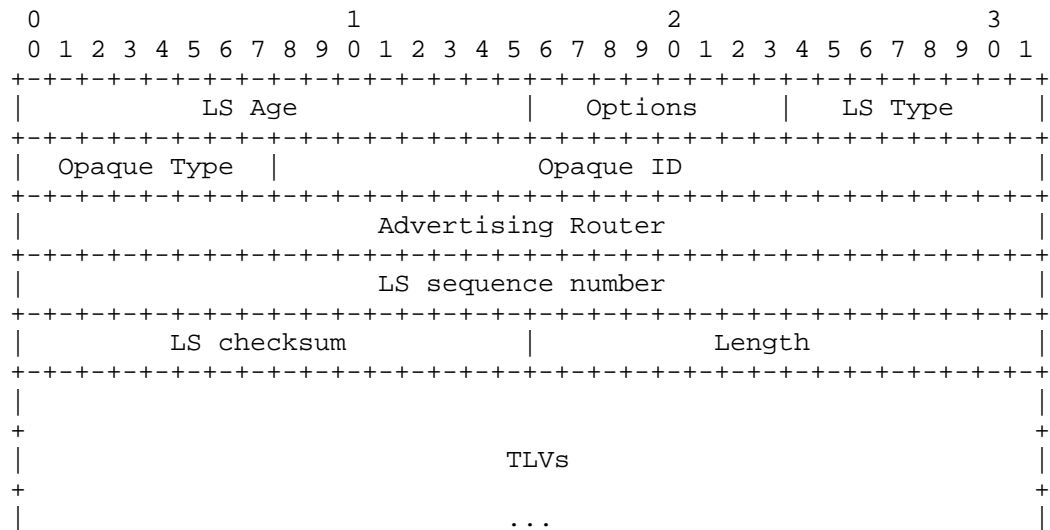


Figure 4: OSPFv2 FlowSpec Opaque LSA

LS age: the same as defined in [RFC2328].

Options: the same as defined in [RFC2328].

LS type: A type-11 or type-10 Opaque-LSA SHOULD be originated. Since the type-11 LSA has the same flooding scope as a type-5 LSA as stated in [RFC5250], it will not be flooded into stub areas or NSSAs (Not-So-Stubby Areas). When stub or NSSA areas are encountered in the scenario of flow spec, we may have to make our choice, either making peace with it and filtering the DoS traffic

at ABRs or generating a new type-10 Opaque-LSA into stub/NSSA areas, which may aggravate the burden of devices in that area.

Opaque type: OSPF FlowSpec Opaque LSA (Type Code: TBD1).

Opaque ID: the same as defined in [RFC5250].

Advertising Router: the same as defined in [RFC2328].

LS sequence number: the same as defined in [RFC2328].

LS checksum: the same as defined in [RFC2328].

Length: the same as defined in [RFC2328].

TLVs: one or more TLVs MAY be included in a FlowSpec Opaque LSA to carry FlowSpec information.

The variable TLVs section consists of one or more nested Type/Length/Value (TLV) tuples. Nested TLVs are also referred to as sub-TLVs. The format of each TLV is shown in Figure 5:

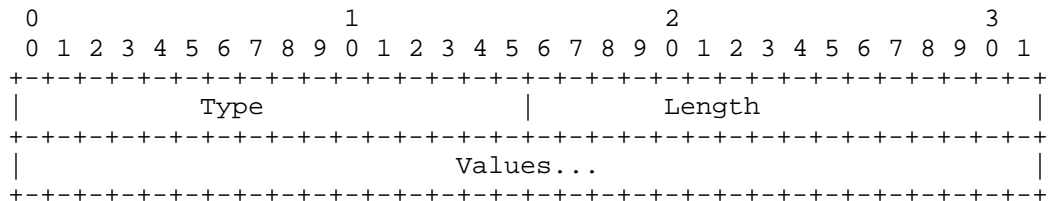


Figure 5: TLV Format

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.

If FlowSpec Opaque LSA is Type-11 Opaque LSA, it is not flooded into Stub and NSSA areas. As the traffic accessing a network segment outside Stub and NSSA areas would be aggregated to the ABR, FlowSpec rules could be applied on the ABR instead of disseminating them into Stub and NSSA areas.

4.1.2. OSPFv3 FlowSpec LSA

This document defines a new OSPFv3 flow specification LSA encoding format that can be used to distribute traffic flow specifications. This new OSPFv3 FlowSpec LSA is extended based on [RFC5340].

The OSPFv3 FlowSpec LSA is defined below in Figure 6:

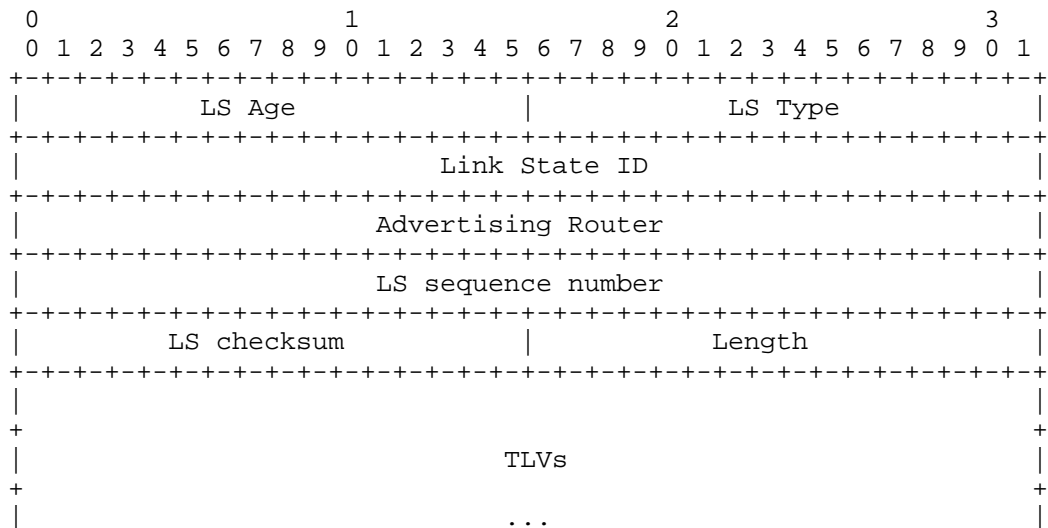


Figure 6: OSPFv3 FlowSpec LSA

LS age: the same as defined in [RFC5340].

LS type: the same as defined in [RFC5340]. The format of the LS type is as follows:

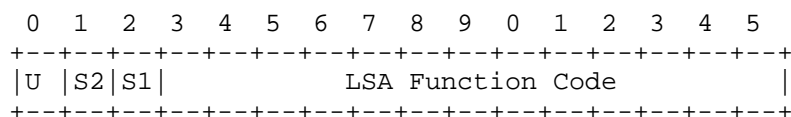


Figure 7: LSA Type

In this document, the U bit should be set indicating that the OSPFv3 FlowSpec LSA should be flooded even if it is not understood. For the area scope, the S1 bit should be set and the S2 should be clear. For the AS scope, the S1 bit should be clear and the S2 bit should be set. A new LSA Function Code (TBD2) needs to be defined for OSPFv3 FlowSpec LSA. To facilitate inter-

area reachability validation, any OSPFv3 router originating AS scoped LSAs is considered an AS Boundary Router (ASBR).

Link State ID: the same as defined in [RFC5340].

Advertising Router: the same as defined in [RFC5340].

LS sequence number: the same as defined in [RFC5340].

LS checksum: the same as defined in [RFC5340].

Length: the same as defined in [RFC5340].

TLVs: one or more TLVs MAY be included in a OSPFv3 FlowSpec LSA to carry FlowSpec information.

4.2. OSPF FlowSpec Filters TLV

The FlowSpec Opaque LSA carries one or more FlowSpec Filters TLVs and corresponding FlowSpec Action TLVs. The OSPF FlowSpec Filters TLV is defined below in Figure 8.

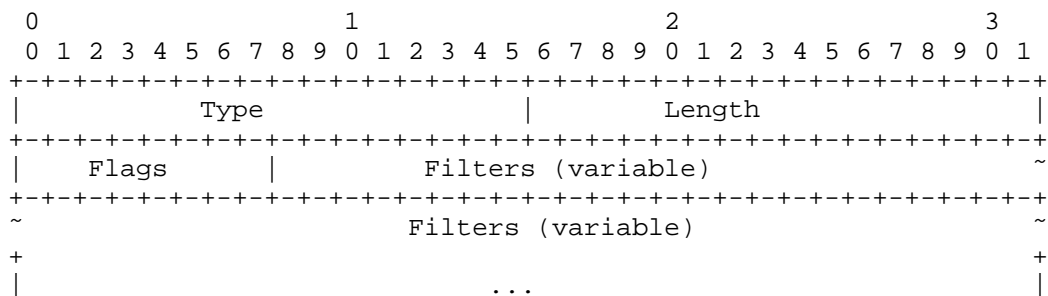
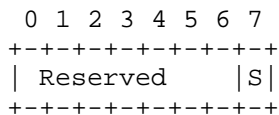


Figure 8: OSPF FlowSpec Filters TLV

Type: the TLV type (Type Code: TBD3)

Length: the size of the value field in octets

Flags: One octet Field identifying Flags.



The least significant bit S is defined as a strict Filter check bit. If set, Strict Validation rules outlined in the validation section Section 4.2.2 need to be enforced.

Filters: the same as "flow-spec NLRI value" defined in [RFC5575] and [I-D.ietf-idr-flow-spec-v6].

Table 1: OSPF Supported FlowSpec Filters

Type	Description	RFC/ WG draft
1	Destination IPv4 Prefix Destination IPv6 Prefix	RFC5575 I-D.ietf-idr-flow-spec-v6
2	Source IPv4 Prefix Source IPv6 Prefix	RFC5575 I-D.ietf-idr-flow-spec-v6
3	IP Protocol Next Header	RFC5575 I-D.ietf-idr-flow-spec-v6
4	Port	RFC5575
5	Destination port	RFC5575
6	Source port	RFC5575
7	ICMP type	RFC5575
8	ICMP code	RFC5575
9	TCP flags	RFC5575
10	Packet length	RFC5575
11	DSCP	RFC5575
12	Fragment	RFC5575
13	Flow Label	I-D.ietf-idr-flow-spec-v6

4.2.1. Order of Traffic Filtering Rules

With traffic filtering rules, more than one rule may match a particular traffic flow. The order of applying the traffic filter rules is the same as described in Section 5.1 of [RFC5575] and in Section 3.1 of [I-D.ietf-idr-flow-spec-v6].

4.2.2. Validation Procedure

[RFC5575] defines a validation procedure for BGP FlowSpec rules, and [I-D.ietf-idr-bgp-flowspec-oid] describes a modification to the validation procedure defined in [RFC5575] for the dissemination of BGP flow specifications. The OSPF FlowSpec should support similar features to mitigate the unnecessary application of traffic filter rules. The OSPF FlowSpec validation procedure is described as follows.

When a router receives a FlowSpec rule including a destination prefix filter from its neighbor router, it should consider the prefix filter as a valid filter unless the S bit in the flags field of Filter TLV is set. If the S bit is set, then the FlowSpec rule is considered valid if and only if:

The originator of the FlowSpec rule matches the originator of the best-match unicast route for the destination prefix embedded in the FlowSpec.

The former rule allows any centralized controller to originate the prefix filter and advertise it within a given OSPF network. The latter rule, also known as a Strict Validation rule, allows strict checking and enforces that the originator of the FlowSpec filter is also the originator of the destination prefix.

When multiple equal-cost paths exist in the routing table entry, each path could end up having a separate set of FlowSpec rules.

When a router receives a FlowSpec rule not including a destination prefix filter from its neighbor router, the validation procedure described above is not applicable.

The FlowSpec filter validation state is used by a speaker when the filter is considered for an installation in its FIB. An OSPF speaker MUST flood OSPF FlowSpec LSA as per the rules defined in [RFC2328] regardless of the validation state of the prefix filters.

4.3. OSPF FlowSpec Action TLV

There are one or more FlowSpec Action TLVs associated with a FlowSpec Filters TLV. Different FlowSpec Filters TLV could have the same FlowSpec Action TLVs. The following OSPF FlowSpec action TLVs, except Redirect, are same as defined in [RFC5575].

Redirect: IPv4 or IPv6 address. This IP address may correspond to a tunnel, i.e., the redirect allows the traffic to be redirected to a directly attached next-hop or a next-hop requiring a route lookup.

Table 2: Traffic Filtering Actions in [RFC5575], etc.

type	FlowSpec Action	RFC/WG draft
0x8006	traffic-rate	RFC5575
0x8007	traffic-action	RFC5575
0x8108	redirect-to-IPv4	I-D.ietf-idr-flowspec-redirect-rt-bis
0x800b	redirect-to-IPv6	I-D.ietf-idr-flow-spec-v6
0x8009	traffic-marking	RFC5575

4.3.1. Traffic-rate

Traffic-rate TLV is encoded as:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| TBD5, 0x8006 suggested | 4 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Traffic-rate |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Traffic-rate: the same as defined in [RFC5575].

4.3.2. Traffic-action

Traffic-action TLV is encoded as:

```

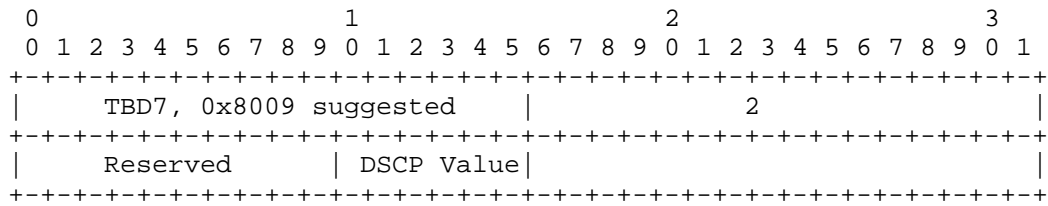
      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| TBD6, 0x8007 suggested | 2 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Reserved | S | T |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

S flag and T flag: the same as defined in [RFC5575].

4.3.3. Traffic-marking

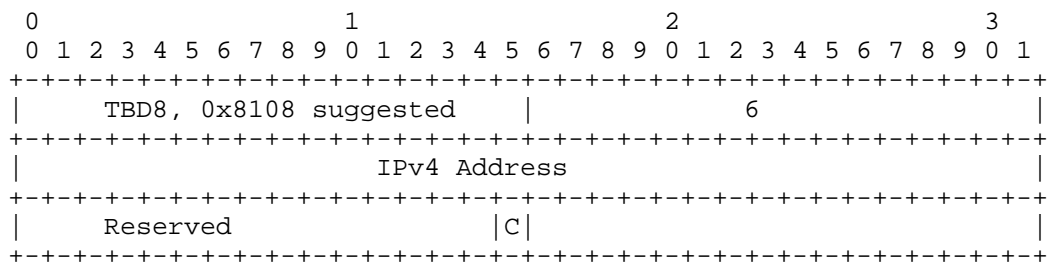
Traffic-marking TLV is encoded as:



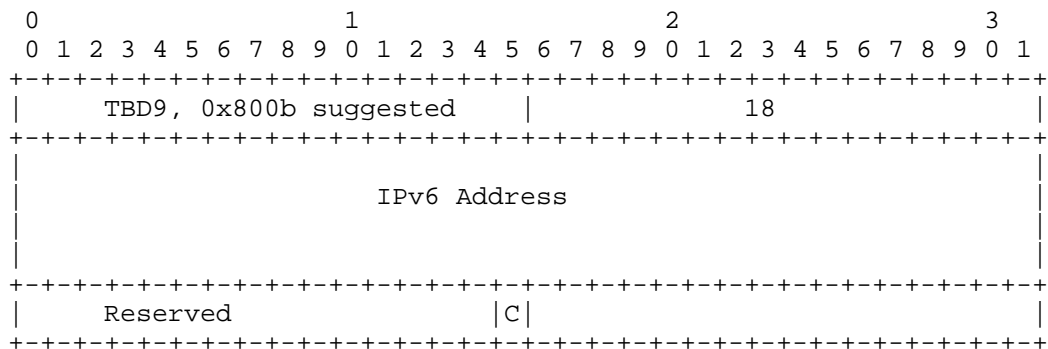
DSCP value: the same as defined in [RFC5575].

4.3.4. Redirect-to-IP

Redirect-to-IPv4 is encoded as:



Redirect to IPv6 TLV is encoded as (Only for OSPFv3):



IPv4/6 Address: the redirection target address.

'C' (or copy) bit: when the 'C' bit is set, the redirection applies to copies of the matching packets and not to the original traffic stream [I-D.ietf-idr-flowspec-redirect-ip].

4.4. Capability Advertisement

This document defines a capability bit for OSPF Router-Information LSA [I-D.ietf-ospf-rfc4970bis] as FlowSpec Capability Advertisement bit. When set, the OSPF router indicates its ability to support the FlowSpec functionality. The FlowSpec Capability Advertisement bit has a value to be assigned by IANA from OSPF Router Functional Capability Bits Registry [I-D.ietf-ospf-rfc4970bis].

5. Redistribution of FlowSpec Routes

In certain scenarios, FlowSpec routes MAY get redistributed from one protocol domain to another; specifically from BGP to OSPF and vice-versa. When redistributed from BGP, the OSPF speaker SHOULD generate an Opaque LSA for the redistributed routes and announce it within an OSPF domain. An implementation MAY provide an option for an OSPF speaker to announce a redistributed FlowSpec route within a OSPF domain regardless of being installed in its local FIB. An implementation MAY impose an upper bound on number of FlowSpec routes that an OSPF router MAY advertise.

6. IANA Considerations

This document defines a new OSPFv2 Opaque LSA, i.e., OSPFv2 FlowSpec Opaque LSA (Type Code: TBD1), which is used to distribute traffic flow specifications.

This document defines a new OSPFv3 LSA, i.e., OSPFv3 FlowSpec LSA (LSA Function Code: TBD2), which is used to distribute traffic flow specifications.

This document defines OSPF FlowSpec Filters TLV (Type Code: TBD3), which is used to describe the filters.

This document defines a new FlowSpec capability which need to be advertised in an RI Opaque LSA. A new informational capability bit needs to be assigned for OSPF FlowSpec feature (FlowSpec Bit: TBD4).

This document defines a new Router LSA bit known as a FlowSpec Capability Advertisement bit. This document requests IANA to assign a bit code type for FlowSpec Capability Advertisement bit from the OSPF Router Functional Capability Bits registry.

Type 1 - Destination IPv4/IPv6 Prefix
Type 2 - Source IPv4/IPv6 Prefix
Type 3 - IP Protocol/Next Header
Type 4 - Port
Type 5 - Destination port
Type 6 - Source port
Type 7 - ICMP type
Type 8 - ICMP code
Type 9 - TCP flags
Type 10 - Packet length
Type 11 - DSCP
Type 12 - Fragment
Type 13 - Flow Label

This document defines a group of FlowSpec actions. The following TLV types need to be assigned:

Type 0x8006(TBD5) - traffic-rate
Type 0x8007(TBD6) - traffic-action
Type 0x8009(TBD7) - traffic-marking
Type 0x8108(TBD8) - redirect to IPv4
Type 0x800b(TBD9) - redirect to IPv6

7. Security considerations

This extension to OSPF does not change the underlying security issues inherent in the existing OSPF. Implementations must assure that malformed TLV and Sub-TLV permutations do not result in errors which cause hard OSPF failures.

8. Acknowledgement

The authors would also like to thank Burjiz Pithawala, Rashmi Shrivastava and Mike Dubrovsky for their contribution to the original version of the document.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.

- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.

9.2. Informative References

- [I-D.ietf-idr-bgp-flowspec-oid]
Uttaro, J., Filsfils, C., Smith, D., Alcaide, J., and P. Mohapatra, "Revised Validation Procedure for BGP Flow Specifications", draft-ietf-idr-bgp-flowspec-oid-02 (work in progress), January 2014.
- [I-D.ietf-idr-flow-spec-v6]
Raszuk, R., Pithawala, B., McPherson, D., and A. Andy, "Dissemination of Flow Specification Rules for IPv6", draft-ietf-idr-flow-spec-v6-06 (work in progress), November 2014.
- [I-D.ietf-idr-flowspec-redirect-ip]
Uttaro, J., Haas, J., Texier, M., Andy, A., Ray, S., Simpson, A., and W. Henderickx, "BGP Flow-Spec Redirect to IP Action", draft-ietf-idr-flowspec-redirect-ip-02 (work in progress), February 2015.

Authors' Addresses

Qiandeng Liang
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: liuweihang@huawei.com

Jianjie You
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: youjianjie@huawei.com

Nan Wu
Huawei

Email: eric.wu@huawei.com

Peng Fan
China Mobile

Email: fanpeng@chinamobile.com

Keyur Patel
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95124 95134
USA

Email: keyupate@cisco.com

Acee Lindem
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95124 95134
USA

Email: acee@cisco.com

Open Shortest Path First IGP
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2015

P. Psenak, Ed.
S. Previdi, Ed.
C. Filsfils
Cisco Systems, Inc.
H. Gredler
Juniper Networks, Inc.
R. Shakir
British Telecom
W. Henderickx
Alcatel-Lucent
J. Tantsura
Ericsson
July 2, 2014

OSPFv3 Extensions for Segment Routing
draft-psenak-ospf-segment-routing-ospfv3-extension-02

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF).

This draft describes the necessary OSPFv3 extensions that need to be introduced for Segment Routing.

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 <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 January 3, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Segment Routing Identifiers	3
2.1. SID/Label sub-TLV	3
3. Segment Routing Capabilities	4
3.1. SR-Algorithm TLV	4
3.2. SID/Label Range TLV	5
4. Prefix SID Identifier	7
4.1. Prefix SID Sub-TLV	7
4.2. SID/Label Binding sub-TLV	11
4.2.1. ERO Metric sub-TLV	13
4.2.2. ERO sub-TLVs	13
5. Adjacency Segment Identifier (Adj-SID)	19
5.1. Adj-SID sub-TLV	20
5.2. LAN Adj-SID Sub-TLV	21
6. Elements of Procedure	23
6.1. Intra-area Segment routing in OSPFv3	23
6.2. Inter-area Segment routing in OSPFv3	24
6.3. SID for External Prefixes	25
6.4. Advertisement of Adj-SID	25
6.4.1. Advertisement of Adj-SID on Point-to-Point Links	25
6.4.2. Adjacency SID on Broadcast or NBMA Interfaces	25
7. IANA Considerations	26
7.1. OSPF Router Information (RI) TLVs Registry	26
7.2. OSPFv3 Extend-LSA sub-TLV registry	26
8. Security Considerations	27
9. Contributors	27
10. Acknowledgements	27
11. References	27
11.1. Normative References	27

11.2. Informative References	27
Authors' Addresses	28

1. Introduction

Segment Routing (SR) allows for a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF). Prefix segments represent an ecmp-aware shortest-path to a prefix (or a node), as per the state of the IGP topology. Adjacency segments represent a hop over a specific adjacency between two nodes in the IGP. A prefix segment is typically a multi-hop path while an adjacency segment, in most of the cases, is a one-hop path. SR's control-plane can be applied to both IPv6 and MPLS data-planes, and do not require any additional signaling (other than the regular IGP). For example, when used in MPLS networks, SR paths do not require any LDP or RSVP-TE signaling. Still, SR can interoperate in the presence of LSPs established with RSVP or LDP .

This draft describes the necessary OSPFv3 extensions that need to be introduced for Segment Routing.

Segment Routing architecture is described in [I-D.filsfils-rtgwg-segment-routing].

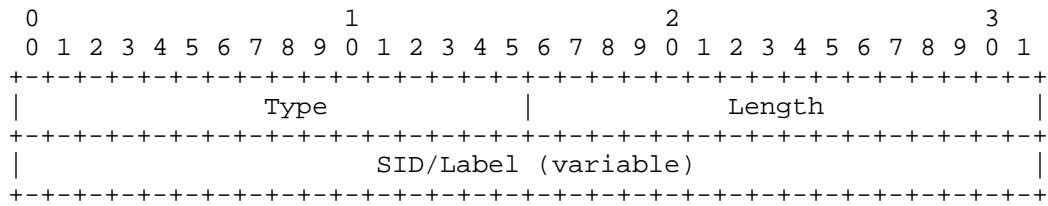
Segment Routing use cases are described in [I-D.filsfils-rtgwg-segment-routing-use-cases].

2. Segment Routing Identifiers

Segment Routing defines various types of Segment Identifiers (SIDs): Prefix-SID, Adjacency-SID, LAN Adjacency SID and Binding SID.

2.1. SID/Label sub-TLV

SID/Label sub-TLV appears in multiple TLVs or Sub-TLVs defined later in this document. It is used to advertise SID or label associated with the prefix or adjacency. SID/Label TLV has following format:



where:

Type: TBD, suggested value 1

Length: variable, 3 or 4 bytes

SID/Label: if length is set to 3, then the 20 rightmost bits represent a label. If length is set to 4 then the value represents a 32 bit SID.

The receiving router MUST ignore SID/Label sub-TLV if the length is other than 3 or 4.

3. Segment Routing Capabilities

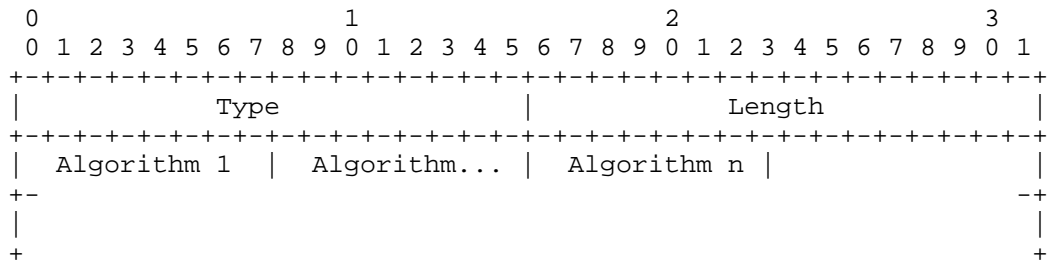
Segment Routing requires some additional capabilities of the router to be advertised to other routers in the area.

These SR capabilities are advertised in OSPFv3 Router Information Opaque LSA (defined in [RFC4970]).

3.1. SR-Algorithm TLV

SR-Algorithm TLV is a TLV of Router Information Opaque LSA (defined in [RFC4970]).

Router may use various algorithms when calculating reachability to other nodes in area or to prefixes attached to these nodes. Examples of these algorithms are metric based Shortest Path First (SPF), various sorts of Constrained SPF, etc. SR-Algorithm TLV allows a router to advertise algorithms that router is currently using to other routers in an area. SR-Algorithm TLV has following structure:



where:

Type: TBD, suggested value 8

Length: variable

Algorithm: one octet identifying the algorithm. The following value has been defined:

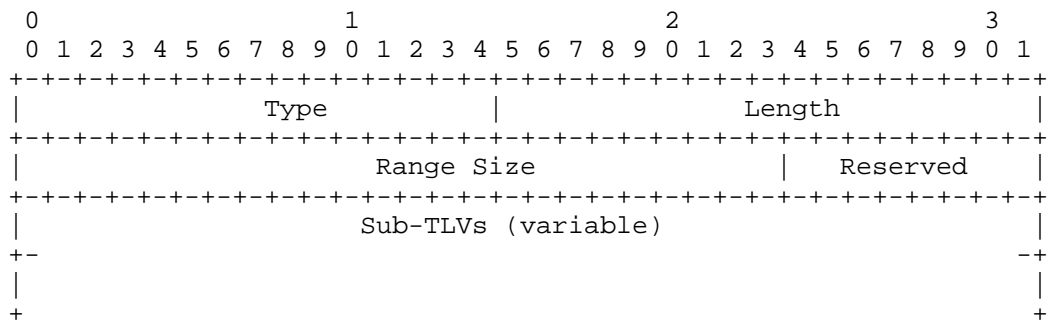
0: IGP metric based SPT.

RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). For the purpose of the SR-Algorithm TLV propagation area scope flooding is required.

3.2. SID/Label Range TLV

The SID/Label Range TLV is a TLV of Router Information Opaque LSA (defined in [RFC4970]).

SID/Label Sub-TLV MAY appear multiple times and has following format:



where:

Type: TBD, suggested value 9

Length: variable

Range Size: 3 octets of SID/label range

Currently the only supported Sub-TLV is the SID/Label TLV as defined in Section 2.1. SID/Label advertised in SID/Label TLV represents the first SID/Label from the advertised range.

Multiple occurrence of the SID/Label Range TLV MAY be advertised, in order to advertise multiple ranges. In such case:

- o The originating router MUST encode each range into a different SID/Label Range TLV.
- o The originating router decides in which order the set of SID/Label Range TLVs are advertised inside Router Information Opaque LSA. The originating router MUST ensure the order is same after a graceful restart (using checkpointing, non-volatile storage or any other mechanism) in order to guarantee the same order before and after graceful restart.
- o Receiving router must adhere to the order in which the ranges are advertised when calculating a SID/label from the SID index.
- o A router not supporting multiple occurrences SID/Label Range TLV MUST take into consideration the first occurrence in the received set.

Here follows an example of advertisement of multiple ranges:

The originating router advertises following ranges:

Range 1: [100, 199]
Range 2: [1000, 1099]
Range 3: [500, 599]

The receiving routers concatenate the ranges and build the SRGB is as follows:

SRGB = [100, 199]
[1000, 1099]
[500, 599]

The indexes span multiple ranges:

index=0 means label 100
...
index 99 means label 199
index 100 means label 1000
index 199 means label 1099
...
index 200 means label 500
...

RI LSA can be advertised at any of the defined flooding scopes (link, area, or autonomous system (AS)). For the purpose of the SR-Capability TLV propagation area scope flooding is required.

4. Prefix SID Identifier

A new extended OSPFv3 LSAs as defined in [I-D.ietf-ospf-ospfv3-lsa-extend] are used to advertise SID or label values associated with the prefix in OSPFv3.

4.1. Prefix SID Sub-TLV

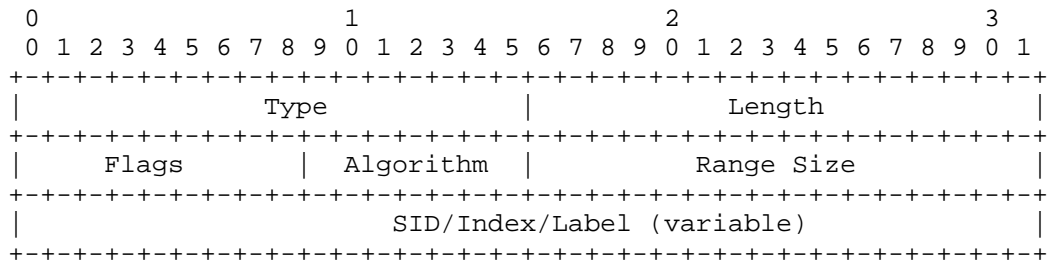
The Prefix SID Sub-TLV is a Sub-TLV of the following OSPFv3 TLVs as defined in [I-D.ietf-ospf-ospfv3-lsa-extend]:

Intra-Area Prefix TLV

Inter-Area Prefix TLV

External Prefix TLV

It MAY appear more than once and has following format:

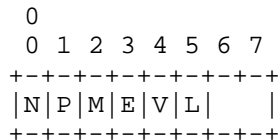


where:

Type: TBD, suggested value 2.

Length: variable

Flags: 1 octet field. The following flags are defined:



where:

N-Flag: Node-SID flag. If set, then the Prefix-SID refers to the router identified by the prefix. Typically, the N-Flag is set on Prefix-SIDs attached to a router loopback address. The N-Flag is set when the Prefix-SID is a Node- SID as described in [I-D.filsfils-rtgwg-segment-routing].

P-Flag: no-PHP flag. If set, then the penultimate hop MUST NOT pop the Prefix-SID before delivering the packet to the node that advertised the Prefix-SID.

M-Flag: Mapping Server Flag. If set, the SID is advertised from the Segment Routing Mapping Server functionality as described in [I-D.filsfils-rtgwg-segment-routing-use-cases].

E-Flag: Explicit-Null Flag. 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) before forwarding the packet.

The V-Flag: Value/Index Flag. If set, then the Prefix-SID carries an absolute value. If not set, then the Prefix-SID carries an index.

The L-Flag: Local/Global Flag. If set, then the value/index carried by the PrefixSID has local significance. If not set, then the value/index carried by this subTLV has global significance.

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

Algorithm: one octet identifying the algorithm the Prefix-SID is associated with as defined in Section 3.1.

Range Size: this field provides the ability to specify a range of addresses and their associated Prefix SIDs. It represents a compression scheme to distribute a continuous Prefix and their continuous, corresponding SID/Label Block. If a single SID is advertised then the Range Size field MUST be set to 1. For range advertisements > 1, Range Size represents the number of addresses that need to be mapped into a Prefix-SID.

SID/Index/Label: label or index value depending on the V-bit setting.

Examples:

A 32 bit global index defining the offset in the SID/Label space advertised by this router - in this case the V and L flags MUST be unset.

A 24 bit local label where the 20 rightmost bits are used for encoding the label value - in this case the V and L flags MUST be set.

If multiple Prefix-SIDs are advertised for the same prefix, the receiving router MUST use the first encoded SID and MAY use the subsequent ones.

When propagating Prefix-SIDs between areas, if multiple prefix-SIDs are advertised for a prefix, an implementation SHOULD preserve the original ordering, when advertising prefix-SIDs to other areas. This allows implementations that only use single Prefix-SID to have a consistent view across areas.

When calculating the outgoing label for the prefix, the router MUST take into account E and P flags advertised by the next-hop router, if next-hop router advertised the SID for the prefix. This MUST be done regardless of next-hop router contributing to the best path to the prefix or not.

P-Flag (no-PHP) MUST be set on the Prefix-SIDs allocated to inter-area prefixes that are originated by the ABR based on intra-area or inter-area reachability between areas. In case the inter-area prefix is generated based on the prefix which is directly attached to the ABR, P-Flag SHOULD NOT be set

P-Flag (no-PHP) MUST NOT be set on the Prefix-SIDs allocated to redistributed prefixes, unless the redistributed prefix is directly attached to ASBR, in which case the P-Flag SHOULD NOT be set.

If the P-flag is not set then any upstream neighbor of the Prefix-SID originator MUST pop the Prefix-SID. This is equivalent to the penultimate hop popping mechanism used in the MPLS dataplane. In such case MPLS EXP bits of the Prefix-SID are not preserved to the ultimate hop (the Prefix-SID being removed). If the P-flag is unset the received E-flag is ignored.

If the P-flag is set then:

If the E-flag is not set then any upstream neighbor of the Prefix-SID originator MUST keep the Prefix-SID on top of the stack. This is useful when the originator of the Prefix-SID must stitch the incoming packet into a continuing MPLS LSP to the final destination. This could occur at an inter-area border router (prefix propagation from one area to another) or at an inter-domain border router (prefix propagation from one domain to another).

If the E-flag is set then any upstream neighbor of the Prefix-SID originator MUST replace the PrefixSID with a Prefix-SID having an Explicit-NULL value. This is useful, e.g., when the originator of the Prefix-SID is the final destination for the related prefix and the originator wishes to receive the packet with the original EXP bits.

When M-Flag is set, P-flag MUST be set and E-bit MUST NOT be set.

Example 1: if the following router addresses (loopback addresses) need to be mapped into the corresponding Prefix SID indexes:

```
Router-A: 192::1/128, Prefix-SID: Index 1
Router-B: 192::2/128, Prefix-SID: Index 2
Router-C: 192::3/128, Prefix-SID: Index 3
Router-D: 192::4/128, Prefix-SID: Index 4
```

then the Address Prefix field in Intra-Area Prefix TLV, Inter-Area Prefix TLV or External Prefix TLV is set to 192::1, Prefix Length in

these TLVs would be set to 128, Range Size in Prefix SID sub-TLV would be set to 4 and Index value would be set to 1.

Example 2: If the following prefixes need to be mapped into the corresponding Prefix-SID indexes:

```
10:1:1::0/120,    Prefix-SID: Index 51
10:1:1::100/120,  Prefix-SID: Index 52
10:1:1::200/120,  Prefix-SID: Index 53
10:1:1::300/120,  Prefix-SID: Index 54
10:1:1::400/120,  Prefix-SID: Index 55
10:1:1::500/120,  Prefix-SID: Index 56
10:1:1::600/120,  Prefix-SID: Index 57
```

then the Address Prefix field in Intra-Area Prefix TLV, Inter-Area Prefix TLV or External Prefix TLV is set to 10:1:1::0, Prefix Length in these TLVs would be set to 120, Range Size in Prefix SID sub-TLV would be set to 7 and Index value would be set to 51.

4.2. SID/Label Binding sub-TLV

SID/Label Binding sub-TLV is used to advertise SID/Label mapping for a path to the prefix.

The SID/Label Binding TLV MAY be originated by any router in an OSPFv3 domain. The router may advertise a SID/Label binding to a FEC along with at least a single 'nexthop style' anchor. The protocol supports more than one 'nexthop style' anchor to be attached to a SID/Label binding, which results into a simple path description language. In analogy to RSVP the terminology for this is called an 'Explicit Route Object' (ERO). Since ERO style path notation allows to anchor SID/label bindings to both link and node IP addresses any label switched path, can be described. Furthermore also SID/Label Bindings from external protocols can get easily re-advertised.

The SID/Label Binding TLV may be used for advertising SID/Label Bindings and their associated Primary and Backup paths. In one single TLV either a primary ERO Path, a backup ERO Path or both are advertised. If a router wants to advertise multiple parallel paths then it can generate several TLVs for the same Prefix/FEC. Each occurrence of a Binding TLV with respect with a given FEC Prefix has accumulating and not canceling semantics.

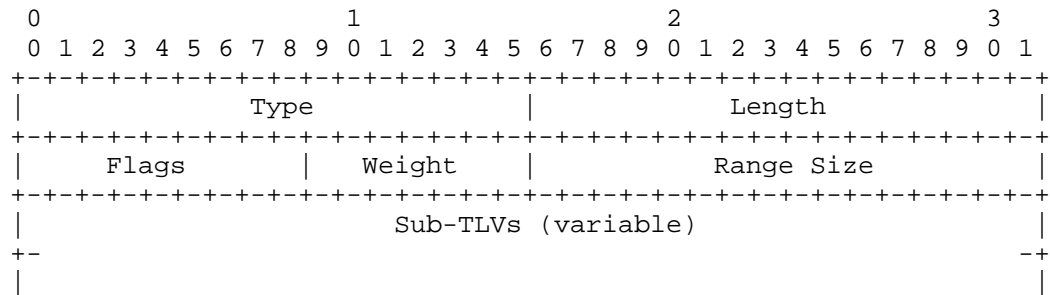
SID/Label Binding sub-TLV is a sub-TLV of the following OSPFv3 TLVs, as defined in [I-D.ietf-ospf-ospfv3-lsa-extend]:

Intra-Area Prefix TLV

Inter-Area Prefix TLV

External Prefix TLV

Multiple SID/Label Binding sub-TLVs can be present in above mentioned TLVs. SID/Label Binding sub-TLV has following format:

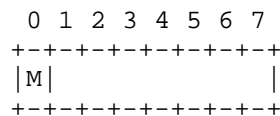


where:

Type: TBD, suggested value 5

Length: variable

Flags: 1 octet field of following flags:



where:

M-bit - When the bit is set the binding represents the mirroring context as defined in [I-D.minto-rsvp-lsp-egress-fast-protection].

Weight: weight used for load-balancing purposes. The use of the weight is defined in [I-D.filesfils-rtgwg-segment-routing].

Range Size: usage is the same as described in Section 4.1

SID/Label Binding sub-TLV currently supports following Sub-TLVs:

SID/Label sub-TLV as described in Section 2.1. This sub-TLV MUST appear in the SID/Label Binding Sub-TLV and it MUST only appear once.

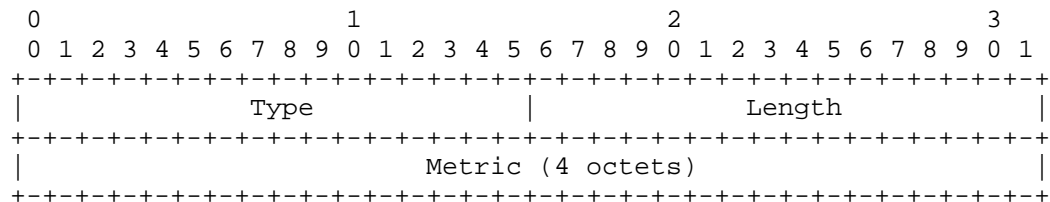
ERO Metric sub-TLV as defined in Section 4.2.1.

ERO sub-TLVs as defined in Section 4.2.2.

4.2.1. ERO Metric sub-TLV

ERO Metric sub-TLV is a Sub-TLV of the SID/Label Binding TLV.

The ERO Metric sub-TLV carries the cost of an ERO path. It is used to compare the cost of a given source/destination path. A router SHOULD advertise the ERO Metric sub-TLV. The cost of the ERO Metric sub-TLV SHOULD be set to the cumulative IGP or TE path cost of the advertised ERO. Since manipulation of the Metric field may attract or distract traffic from and to the advertised segment it MAY be manually overridden.



ERO Metric sub-TLV format

where:

Type: TBD, suggested value 6

Length: 4 bytes

Metric: 4 bytes

4.2.2. ERO sub-TLVs

All 'ERO' information represents an ordered set which describes the segments of a path. The last ERO sub-TLV describes the segment closest to the egress point, contrary the first ERO sub-TLV describes the first segment of a path. If a router extends or stitches a path it MUST prepend the new segments path information to the ERO list.

The above similarly applies to backup EROs.

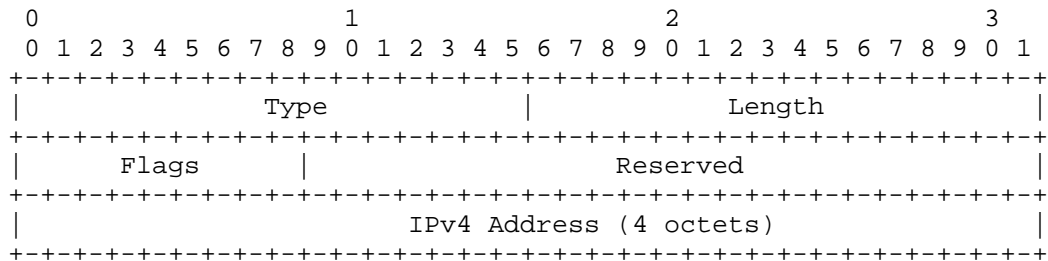
All ERO Sub-TLVs must immediately follow the (SID)/Label Sub-TLV.

All Backup ERO sub-TLVs must immediately follow last ERO Sub-TLV.

4.2.2.1. IPv4 ERO sub-TLV

IPv4 ERO sub-TLV is a sub-TLV of the SID/Label Binding sub-TLV.

The IPv4 ERO sub-TLV describes a path segment using IPv4 Address style of encoding. Its semantics have been borrowed from [RFC3209].



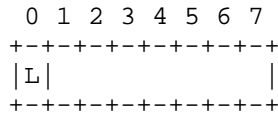
IPv4 ERO sub-TLV format

where:

Type: TBD, suggested value 7

Length: 8 bytes

Flags: 1 octet field of following flags:



where:

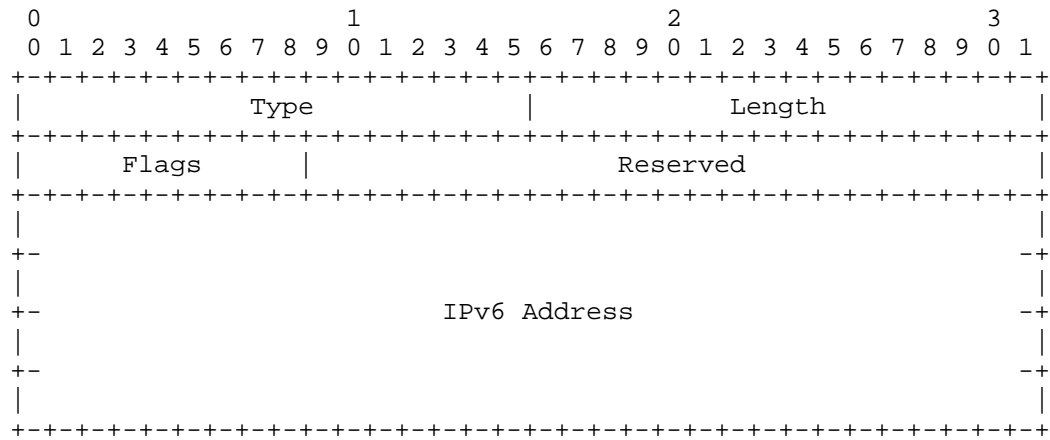
L-bit - If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.'

IPv4 Address - the address of the explicit route hop.

4.2.2.2. IPv6 ERO sub-TLV

IPv6 ERO sub-TLV is a sub-TLV of the SID/Label Binding sub-TLV.

The IPv6 ERO sub-TLV (Type TBA) describes a path segment using IPv6 Address style of encoding. Its semantics have been borrowed from [RFC3209].



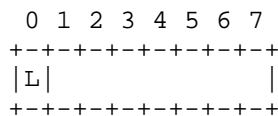
IPv6 ERO sub-TLV format

where:

Type: TBD, suggested value 8

Length: 8 bytes

Flags: 1 octet field of following flags:



where:

L-bit - If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.'

IPv6 Address - the address of the explicit route hop.

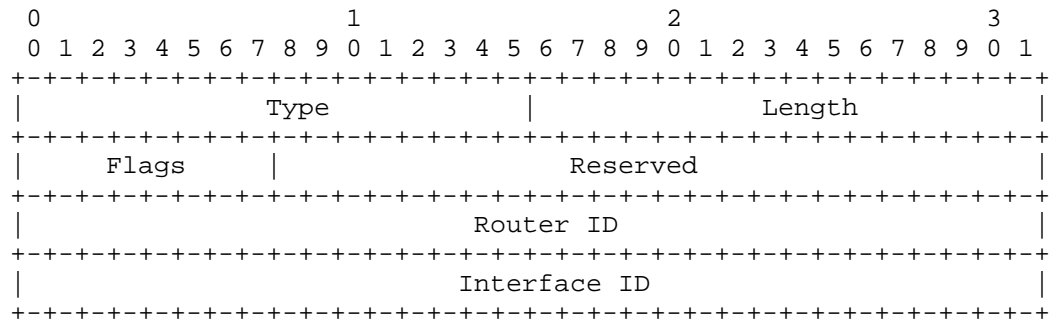
4.2.2.3. Unnumbered Interface ID ERO sub-TLV

Unnumbered Interface ID ERO sub-TLV is a sub-TLV of the SID/Label Binding sub-TLV.

The appearance and semantics of the 'Unnumbered Interface ID' have been borrowed from [RFC3477].

The Unnumbered Interface-ID ERO sub-TLV describes a path segment that spans over an unnumbered interface. Unnumbered interfaces are

referenced using the interface index. Interface indices are assigned local to the router and therefore not unique within a domain. All elements in an ERO path need to be unique within a domain and hence need to be disambiguated using a domain unique Router-ID.



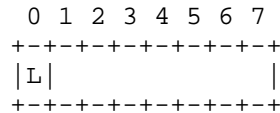
where:

Unnumbered Interface ID ERO sub-TLV format

Type: TBD, suggested value 9

Length: 12 bytes

Flags: 1 octet field of following flags:



where:

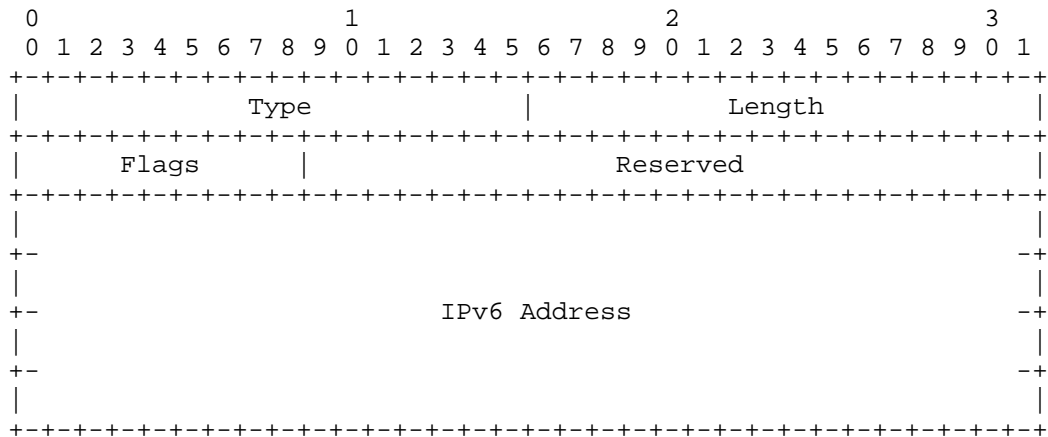
L-bit - If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.'

Router-ID: Router-ID of the next-hop.

Interface ID: is the identifier assigned to the link by the router specified by the Router-ID.

4.2.2.4. IPv4 Backup ERO sub-TLV

IPv4 Prefix Backup ERO sub-TLV is a sub-TLV of the SID/Label Binding sub-TLV.



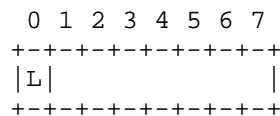
IPv6 Backup ERO sub-TLV format

where:

Type: TBD, suggested value 11

Length: 8 bytes

Flags: 1 octet field of following flags:



where:

L-bit - If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.'

IPv6 Address - the address of the explicit route hop.

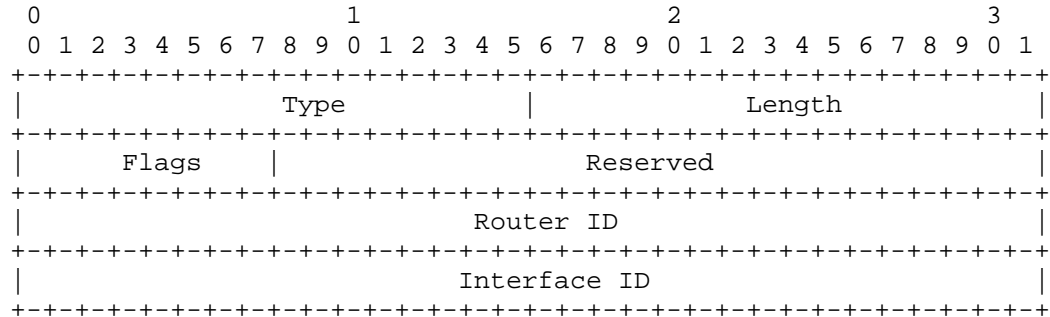
4.2.2.6. Unnumbered Interface ID Backup ERO sub-TLV

Unnumbered Interface ID Backup sub-TLV is a sub-TLV of the SID/Label Binding sub-TLV.

The appearance and semantics of the 'Unnumbered Interface ID' have been borrowed from [RFC3477].

The Unnumbered Interface-ID ERO sub-TLV describes a path segment that spans over an unnumbered interface. Unnumbered interfaces are

referenced using the interface index. Interface indices are assigned local to the router and therefore not unique within a domain. All elements in an ERO path need to be unique within a domain and hence need to be disambiguated using a domain unique Router-ID.



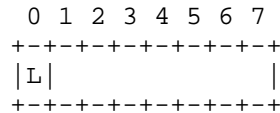
Unnumbered Interface ID Backup ERO sub-TLV format

where:

Type: TBD, suggested value 12

Length: 12 bytes

Flags: 1 octet field of following flags:



where:

L-bit - If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.'

Router-ID: Router-ID of the next-hop.

Interface ID: is the identifier assigned to the link by the router specified by the Router-ID.

5. Adjacency Segment Identifier (Adj-SID)

An Adjacency Segment Identifier (Adj-SID) represents a router adjacency in Segment Routing. At the current stage of Segment Routing architecture it is assumed that the Adj-SID value has local significance (to the router).

5.1. Adj-SID sub-TLV

A new extended OSPFv3 LSAs, as defined in [I-D.ietf-ospf-ospfv3-lsa-extend], are used to advertise prefix SID in OSPFv3

Adj-SID sub-TLV is an optional sub-TLV of the Router-Link TLV as defined in [I-D.ietf-ospf-ospfv3-lsa-extend]. It MAY appear multiple times in Router-Link TLV. Examples where more than one Adj-SID may be used per neighbor are described in [I-D.filsfils-rtgwg-segment-routing-use-cases]. The structure of the Adj-SID Sub-TLV is as follows:

0										1										2										3																			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																		
Type										Length																																							
Flags										Weight										Reserved																													
SID/Label/Index (variable)																																																	

where:

Type: TBD, suggested value 10.

Length: variable.

Flags. 1 octet field of following flags:

0	1	2	3	4	5	6	7
B	V	L	S				

where:

B-Flag: Backup-flag: set if the Adj-SID refer to an adjacency being protected (e.g.: using IPFRR or MPLS-FRR) as described in [I-D.filsfils-rtgwg-segment-routing-use-cases].

The V-Flag: Value/Index Flag. If set, then the Prefix-SID carries an absolute value. If not set, then the Prefix-SID carries an index.

The L-Flag: Local/Global Flag. If set, then the value/index carried by the PrefixSID has local significance. If not set,

then the value/index carried by this subTLV has global significance.

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

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

Weight: weight used for load-balancing purposes. The use of the weight is defined in [I-D.filsfils-rtgwg-segment-routing].

SID/Index/Label: label or index value depending on the V-bit setting.

Examples:

A 32 bit global index defining the offset in the SID/Label space advertised by this router - in this case the V and L flags MUST be unset.

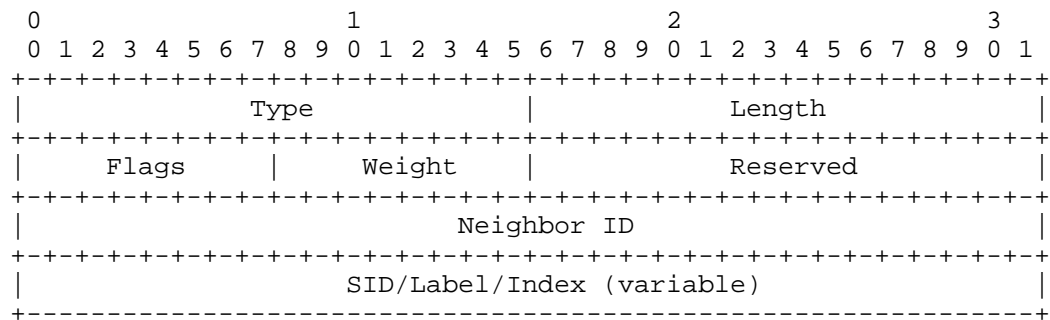
A 24 bit local label where the 20 rightmost bits are used for encoding the label value - in this case the V and L flags MUST be set.

16 octet IPv6 address - in this case the V-flag MUST be set. The L-flag MUST be set for link-local IPv6 address and MUST be unset for IPv6 global unicast address.

A SR capable router MAY allocate an Adj-SID for each of its adjacencies and set the B-Flag when the adjacency is protected by a FRR mechanism (IP or MPLS) as described in [I-D.filsfils-rtgwg-segment-routing-use-cases].

5.2. LAN Adj-SID Sub-TLV

LAN Adj-SID is an optional sub-TLV of the Router-Link TLV. It MAY appear multiple times in Router-Link TLV. It is used to advertise SID/Label for adjacency to non-DR node on broadcast or NBMA network.

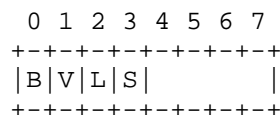


where:

Type: TBD, suggested value 11.

Length: variable.

Flags. 1 octet field of following flags:



where:

B-Flag: Backup-flag: set if the LAN-Adj-SID refer to an adjacency being protected (e.g.: using IPFRR or MPLS-FRR) as described in [I-D.filsfils-rtgwg-segment-routing-use-cases].

The V-Flag: Value/Index Flag. If set, then the Prefix-SID carries an absolute value. If not set, then the Prefix-SID carries an index.

The L-Flag: Local/Global Flag. If set, then the value/index carried by the PrefixSID has local significance. If not set, then the value/index carried by this subTLV has global significance.

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

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

Weight: weight used for load-balancing purposes. The use of the weight is defined in [I-D.filsfils-rtgwg-segment-routing].

SID/Index/Label: label or index value depending on the V-bit setting.

Examples:

A 32 bit global index defining the offset in the SID/Label space advertised by this router - in this case the V and L flags MUST be unset.

A 24 bit local label where the 20 rightmost bits are used for encoding the label value - in this case the V and L flags MUST be set.

16 octet IPv6 address - in this case the V-flag MUST be set. The L-flag MUST be set for link-local IPv6 address and MUST be unset for IPv6 global unicast address.

6. Elements of Procedure

6.1. Intra-area Segment routing in OSPFv3

The OSPFv3 node that supports segment routing MAY advertise Prefix-SIDs for any prefix that it is advertising reachability for (e.g. loopback IP address) as described in Section 4.1.

If multiple routers advertise Prefix-SID for the same prefix, then the Prefix-SID MUST be the same. This is required in order to allow traffic load-balancing if multiple equal cost paths to the destination exist in the network.

Prefix-SID can also be advertised by the SR Mapping Servers (as described in [I-D.filsfils-rtgwg-segment-routing-use-cases]). The Mapping Server advertises Prefix-SID for remote prefixes that exist in the network. Multiple Mapping Servers can advertise Prefix-SID for the same prefix, in which case the same Prefix-SID MUST be advertised by all of them. SR Mapping Server could use either area scope or autonomous system flooding scope when advertising Prefix SID for prefixes, based on the configuration of the SR Mapping Server. Depending on the flooding scope used, SR Mapping Server chooses the LSA that will be used. If the area flooding scope is needed, E-Intra-Area-Prefix-LSA ([I-D.ietf-ospf-ospfv3-lsa-extend]) is used. If autonomous system flooding scope is needed, E-AS-External-LSA ([I-D.ietf-ospf-ospfv3-lsa-extend]) is used.

When Prefix-SID is advertised by the Mapping Server, which is indicated by the M-flag in the Prefix-SID sub-TLV (Section 4.1), route-type as indicated by the LSA type which is being used for flooding is ignored. Prefix SID is bound to a prefix, in which case route-type becomes unimportant.

Advertisement of the Prefix-SID by the Mapping Server using Inter-Area Prefix TLV, External Prefix TLV or Intra-Area-Prefix TLV ([I-D.ietf-ospf-ospfv3-lsa-extend]) does not itself contribute to the prefix reachability. NU-bit MUST be set in the PrefixOptions field of the LSA which is used by the Mapping Server to advertise SID or SID range, which prevents such advertisement to contribute to the prefix reachability.

6.2. Inter-area Segment routing in OSPFv3

In order to support SR in a multi-area environment, OSPFv3 must propagate Prefix-SID information between areas. The following procedure is used in order to propagate Prefix SIDs between areas.

When an OSPFv3 ABR advertises a Inter-Area-Prefix-LSA from an intra-area prefix to all its connected areas, it will also include Prefix-SID sub-TLV, as described in Section 4.1. The Prefix-SID value will be set as follows:

The ABR will look at its best path to the prefix in the source area and find out the advertising router associated with its best path to that prefix.

If no Prefix-SID was advertised for the prefix in the source area by the router that contributes to the best path to the prefix, then the ABR will use the Prefix-SID advertised by any other router (e.g.: a Prefix-SID coming from an SR Mapping Server as defined in [I-D.filsfils-rtgwg-segment-routing-use-cases]) when propagating Prefix-SID for the prefix to other areas.

When an OSPFv3 ABR advertises Inter-Area-Prefix-LSA LSAs from an inter-area route to all its connected areas it will also include Prefix-SID sub-TLV, as described in Section 4.1. The Prefix-SID value will be set as follows:

The ABR will look at its best path to the prefix in the source area and find out the advertising router associated with its best path to that prefix.

The ABR will then look if such router advertised a Prefix-SID for the prefix and use it when advertising the Prefix-SID to other connected areas.

If no Prefix-SID was advertised for the prefix in the source area by the ABR that contributes to the best path to the prefix, the originating ABR will use the Prefix-SID advertised by any other router (e.g.: a Prefix-SID coming from an SR Mapping Server as defined in [I-D.filsfils-rtgwg-segment-routing-use-cases]) when propagating Prefix-SID for the prefix to other areas.

6.3. SID for External Prefixes

AS-External-LSAs are flooded domain wide. When an ASBR, which supports SR, generates AS-External-LSA, it should also include Prefix-SID sub-TLV, as described in Section 4.1 Prefix-SID value will be set to the SID that has been reserved for that prefix.

When a NSSA ASBR translates NSSA-LSA into AS-External-LSA, it should also advertise the Prefix-SID for the prefix. The NSSA ABR determines its best path to the prefix advertised in the translated NSSA-LSA and finds the advertising router associated with such path. If such advertising router has advertised a Prefix-SID for the prefix, then the NSSA ASBR uses it when advertising the Prefix-SID in AS-External-LSA. Otherwise the Prefix-SID advertised by any other router will be used (e.g.: a Prefix-SID coming from an SR Mapping Server as defined in [I-D.filsfils-rtgwg-segment-routing-use-cases]).

6.4. Advertisement of Adj-SID

The Adjacency Segment Routing Identifier (Adj-SID) is advertised using the Adj-SID Sub-TLV as described in Section 5.

6.4.1. Advertisement of Adj-SID on Point-to-Point Links

Adj-SID MAY be advertised for any adjacency on p2p link that is in a state 2-Way or higher. If the adjacency on a p2p link transitions from the FULL state, then the Adj-SID for that adjacency MAY be removed from the area. If the adjacency transitions to a state lower than 2-Way, then the Adj-SID MUST be removed from the area.

6.4.2. Adjacency SID on Broadcast or NBMA Interfaces

Broadcast or NBMA networks in OSPFv3 are represented by a star topology where the Designated Router (DR) is the central point all other routers on the broadcast or NBMA network connect to. As a result, routers on the broadcast or NBMA network advertise only their adjacency to DR and BDR. Routers that are neither DR nor BDR do not form and do not advertise adjacencies between them. They, however, maintain a 2-Way adjacency state between them.

When Segment Routing is used, each router on the broadcast or NBMA network MAY advertise the Adj-SID for its adjacency to DR using Adj-SID Sub-TLV as described in Section 5.1.

SR capable router MAY also advertise Adj-SID for other neighbors (e.g. BDR, DR-OTHER) on broadcast or NBMA network using the LAN ADJ-SID Sub-TLV as described in section 5.1.1.2. Section 5.2.

7. IANA Considerations

This specification updates two existing OSPF registries.

7.1. OSPF Router Information (RI) TLVs Registry

- o suggested value 8 - SR-Algorithm TLV
- o suggested value 9 - SID/Label Range TLV

7.2. OSPFv3 Extend-LSA sub-TLV registry

- o suggested value 1 - SID/Label sub-TLV
- o suggested value 2 - Prefix SID sub-TLV
- o suggested value 3 - Adj-SID sub-TLV
- o suggested value 4 - LAN Adj-SID sub-TLV
- o suggested value 5 - SID/Label Binding sub-TLV
- o suggested value 6 - ERO Metric sub-TLV
- o suggested value 7 - IPv4 ERO sub-TLV
- o suggested value 8 - IPv6 ERO sub-TLV
- o suggested value 9 - Unnumbered Interface ID ERO sub-TLV
- o suggested value 10 - IPv4 Backup ERO sub-TLV
- o suggested value 11 - IPv6 Backup ERO sub-TLV
- o suggested value 12 - Unnumbered Interface ID Backup ERO sub-TLV

8. Security Considerations

Implementations must assure that malformed permutations of the newly defined sub-TLVs do not result in errors which cause hard OSPFv3 failures.

9. Contributors

The following people gave a substantial contribution to the content of this document: Ahmed Bashandy, Martin Horneffer, Bruno Decraene, Stephane Litkowski, Igor Milojevic, Rob Shakir and Saku Ytti.

10. Acknowledgements

We would like to thank Anton Smirnov for his contribution.

Many thanks to Yakov Rekhter, John Drake and Shraddha Hedge for their contribution on earlier incarnations of the "Binding / MPLS Label TLV" in [I-D.gredler-ospf-label-advertisement].

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.
- [RFC3477] Kompella, K. and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, January 2003.
- [RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.

11.2. Informative References

- [I-D.filsfils-rtgwg-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-filsfils-rtgwg-segment-routing-01 (work in progress), October 2013.

[I-D.filsfils-rtgwg-segment-routing-use-cases]

Filsfils, C., Francois, P., Previdi, S., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., Kini, S., and E. Crabbe, "Segment Routing Use Cases", draft-filsfils-rtgwg-segment-routing-use-cases-02 (work in progress), October 2013.

[I-D.gredler-ospf-label-advertisement]

Gredler, H., Amante, S., Scholl, T., and L. Jalil, "Advertising MPLS labels in OSPF", draft-gredler-ospf-label-advertisement-03 (work in progress), May 2013.

[I-D.ietf-ospf-ospfv3-lsa-extend]

Lindem, A., Mirtorabi, S., Roy, A., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-03 (work in progress), May 2014.

[I-D.minto-rsvp-lsp-egress-fast-protection]

Jeganathan, J., Gredler, H., and Y. Shen, "RSVP-TE LSP egress fast-protection", draft-minto-rsvp-lsp-egress-fast-protection-03 (work in progress), November 2013.

Authors' Addresses

Peter Psenak (editor)
Cisco Systems, Inc.
Apollo Business Center
Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: sprevidi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Hannes Gredler
Juniper Networks, Inc.
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: hannes@juniper.net

Rob Shakir
British Telecom
London
UK

Email: rob.shakir@bt.com

Wim Henderickx
Alcatel-Lucent
Copernicuslaan 50
Antwerp 2018
BE

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson
300 Holger Way
San Jose, CA 95134
US

Email: Jeff.Tantsura@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2016

K. Raza
vIPtela
J. Cavanaugh
405Labs
A. Kulawiak
Morgan Stanley
P. Pillay-Esnault
F. Shamim
Cisco Systems
October 18, 2015

OSPF Stub Neighbors
draft-raza-ospf-stub-neighbor-02

Abstract

Open Shortest Path First stub neighbor is an enhancement to the protocol to support large scale of neighbors in some topologies with improved convergence behavior. It introduces limited changes protocol behavior to implement a scalable solution for hub and spoke topologies by limiting the functionality changes to the hub. The concepts are also applicable to a host running in a virtual machine environment.

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 April 20, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Specification of Requirements	4
3. Incremental deployment	4
4. Link State Advertisement Filtering	4
4.1. Area Border Router(ABR) Hub Routers	5
4.2. Autonomous System Boundary Router (ASBR) Hub Routers	5
4.3. Hub Routers which are neither ASBR or ABR	5
5. Proposed Changes	5
5.1. Stub neighbor overview	5
5.2. Local Adjacency	5
5.3. Local Router LSA originated on the Hub Router	6
6. Hub Router Stub Neighbor Support Discovery and misconfiguration detection	8
7. Receiving and propagation of spoke routes	10
8. Demand Circuit	10
9. Benefits	10
10. Security Considerations	11
11. IANA Considerations	11
12. Acknowledgments	11
13. Normative References	11
Authors' Addresses	11

1. Introduction

With the growing size of an OSPF-network, most large networks are now deploying OSPF in large hub and spoke topologies. Also in lot of cases L3 routing would be extended to Top of rack or even to a host running virtual machines.

In any case these remote devices constitute a stub point in an OSPF network. These devices although being part of OSPF network will never be a transit point and thus do not need any topology information of the area nor do they require optimal routing calculations.

The spoke router in the case of a hub and spoke (or a host running OSPF) only need default route to the rest of the network, but they do need to send information about the connected network in the local site. In case of hosts they need to advertise routes in the virtual machines.

OSPF as network protocol was designed for an environment where routers were of similar capabilities. To protect the larger network, area hierarchy was introduced. Network was typically broken up into a backbone area and several subordinate areas. This breakup of the topology into areas serves multiple purposes

As OSPF has become pervasive protocol in the enterprise network it needs to evolve for large hub and spoke setups, these are typical retail environments. In a retail setup typical remote branch router does not have enough capacity to become part of a larger area, even if we break the network in large number of smaller areas. A remote router in one retail store does not need to have routes to all the router in other retail store that are part of its area setup.

Also increasing the number of areas on ABR can burden the ABR, this is due to the creation of large number of summary LSA. Although this can be handled by creating the areas as stub with no summary. Even by creating smaller sized areas with stub no summary, it does not completely eliminate the problem of having unnecessary information from the prospective of intra area.

With the advent of virtualized hosts, hosts are now advertising an increasing number of new virtual machine routes. These prefixes need to be advertised by a router that is connected to the host. Traditionally the host would be connected to the router via a shared link between the two (host and router). The host is often sourcing subnets that are not connected to the common subnet between the host and routers. However, the hosts (or spokes) themselves just need a default route from the router (or hub) to reach rest of the network. The solutions using current features of the protocol are not scalable. The overhead of protocol info and flooding of large number of unnecessary information to low-end routers caps the number of spokes on a hub.

This document describes extensions to OSPF to support very large Hub and spoke topologies more efficiently. Currently, the spoke router receives unnecessary information from the neighboring hub routers about all the other routers in the area. In most cases all a spoke router needs is IP reachability to hub routers which are the gateways to the rest of the network.

We presuppose familiarity with the contents of [RFC2328].

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 [RFC2119].

3. Incremental deployment

For ease of deployment, the changes proposed in this document will be limited to the hub routers only.

By limiting changes only to the hub router the feature can be incrementally introduced without upgrading other routers in the network. Specifically, the spoke sites do not need to be upgraded.

It will be the responsibility of the hub router to mask the changes from the spoke as well as rest of the OSPF network such that the upgrading the network is simple from the point of interoperability and ease of deployment.

The hub router can be a normal router and there is no requirement for the hub to be a area border router or an autonomous system boundary router. Hub site is a sort of passive listener. It is there to receive routes from the spoke site, and to just provide exit towards rest of the network. A hub router SHOULD send a default or aggregated route towards the spoke and filters out all the information about rest of the network from the spoke.

4. Link State Advertisement Filtering

Routers establish adjacencies to flood topological information. The flooding process ensures all the information is consistent across the entire area and ensures the LSAs are delivered to all routers within the same area.

From the protocol prospective, topological information that is carried in the LSAs cannot be filtered, which it is essential to the loop free topology.

The topological information learned, by all routers within an area build the consistent graph of the network connections.

Vendors have implemented LSA filtering function on per neighbors basis specially for the purpose of scaling large full mesh environments. ISIS had the concept of mesh groups to avoid n2 flooding for a link failure and n3 flooding issue in case of node failure. LSA filtering gives the capability to filter information since it was done in the past in meshed topologies it was very

crucial that planning is done to make sure inconsistency does not happen inside of database thus causing loops.

Today prefix aggregation can only be achieved using summary type 3 or type 5 LSA. There is no way to limit or mask intra area information. The hub and spoke topologies or Data center cases, it would be beneficial to mask intra area information as it would not cause any loop.

4.1. Area Border Router(ABR) Hub Routers

In the case of hub routers being area border routers, aggregation can be achieved at the Hub router level using current features. The aggregation can be done by either using ranges or the default route injected as a type 3 LSA.

4.2. Autonomous System Boundary Router (ASBR) Hub Routers

In the case of hub routers being ASBR as well , aggregation can be achieved at the Hub router level using current features. The aggregation can be done by either using ranges or the default route injected as a type 5 or type 7 LSA.

4.3. Hub Routers which are neither ASBR or ABR

Currently there is no possibility of aggregating prefixes sent to the spoke routers and severely impact the scale.

5. Proposed Changes

5.1. Stub neighbor overview

We propose a new kind of adjacency for neighbors configured as stub. This adjacency will have a modified flooding content as the stub router only need a gateway through its neighbor. The hub router will send limited information to the remote spoke router without overwhelming the host with area topology. Another benefit is failures of the spoke node will be masked and would not impact the larger OSPF domain and other spoke nodes in the network. Spoke nodes SHOULD be considered a stub node when the remote site needs to send only prefixes to rest of the OSPF network without being considered a transit node.

5.2. Local Adjacency

The local adjacency concept is only present on a Hub router and it applies to those neighbors configured as stub neighbors. In this case, the hub router will maintain the adjacency to stub neighbors as

local only. Local adjacencies are not advertised in the normal router LSA flooded to other non-stub neighbors, thus masking the local adjacencies or stub nodes.

On the other hand, the hub router will flood a simplified router LSA to its local adjacencies so as to mask the area topology behind it. The Hub "Local" router LSA will contain only a p2p link to the stub neighbor when full adjacency is achieved and advertise one stub link with a configured range or the default prefix or both. The Hub router will effectively hide all the area topology including the prefixes behind it.

We are introducing a new type of default route with a local behavior. The current use of default route as type 3 or as type 5 cannot solve some of the use cases and more specifically in the Data center topologies.

The spoke router will function as normal advertising all its connected prefixes to Hub router.

5.3. Local Router LSA originated on the Hub Router

The local Router LSA MUST contain at least 2 links. One p2p link to the stub neighbor and a stub link to advertise the default prefix or a range defined per configuration.

```

Hub router-LSA for any area with default prefix
LS age = 0                                ;always true on origination
Options =                                ;
LS type = 1                               ;indicates router-LSA
Link State ID = 192.0.2.1                 ;Hub Router ID
Advertising Router = 192.0.2.1           ;Hub Router ID
bit E = 0                                ;not an AS boundary router
bit B = 0                                ;not area border router
#links = 2
Link ID = 192.0.2.2                       ;Spoke Router ID.
Link Data = 192.0.2.1                     ;Hub IP interface to net
Type = 1                                  ;connects to Point-to-point network
# TOS metrics = 0
metric = 1

Link ID = 0.0.0.0                         ;Default prefix
Link Data = 0x0                           ;Network mask
Type = 3                                  ;connects to stub network
# TOS metrics = 0
metric = 100

```

Hub router-LSA for any area with default prefix

```
Hub router-LSA for any area with configured ranges
LS age = 0                                ;always true on origination
Options =                                ;
LS type = 1                              ;indicates router-LSA
Link State ID = 192.0.2.1                ;Hub Router ID
Advertising Router = 192.0.2.1          ;Hub Router ID
bit E = 0                                ;not an AS boundary router
bit B = 0                                ;not area border router
#links = 2
Link ID = 192.0.2.2                      ;Spoke Router ID.
Link Data = 192.0.2.1                   ;Hub interface to net
Type = 1                                ;connects to Point-to-point network
# TOS metrics = 0
metric = 1

Link ID = 198.51.100.0                  ;Aggregated prefix
Link Data = 0xffffffff00                ;Network mask
Type = 3                                ;connects to stub network
# TOS metrics = 0
metric = 100
```

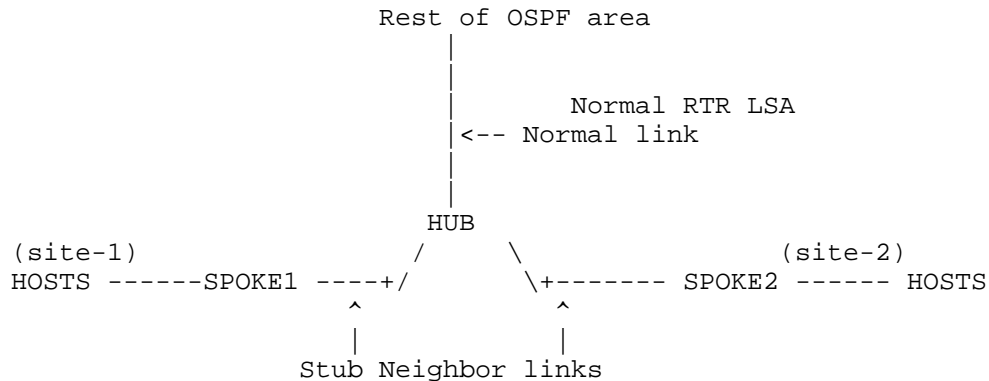
Hub router-LSA for any area with configured ranges

```
Hub router-LSA for any area with configured ranges
LS age = 0                                ;always true on origination
Options =                                ;
LS type = 1                              ;indicates router-LSA
Link State ID = 192.0.2.1                ;Hub Router ID
Advertising Router = 192.0.2.1          ;Hub Router ID
bit E = 0                                ;not an AS boundary router
bit B = 0                                ;not area border router
#links = 2
Link ID = 192.0.2.2                      ;Spoke Router ID.
Link Data = 192.0.2.1                   ;Hub interface to net
Type = 1                                ;connects to Point-to-point network
# TOS metrics = 0
metric = 1

Link ID = 0.0.0.0                       ;Default prefix
Link Data = 0x0                         ;Network mask
Type = 3                                ;connects to stub network
# TOS metrics = 0
metric = 100
```

Hub router-LSA for any area with configured ranges

A spoke router is usually a leaf node or in some cases may be in a dual-homed topology with another hub. In these cases, both Hub routers MUST be configured to view the spoke as a stub neighbor. The Local Router LSA of a Hub will get flooded over the other ospf interfaces of a spoke router. The Hub routers SHOULD ignore local router LSAs from other Hub routers flooded by a stub neighbor.



Simplified HUB Local RTR LSA contains only p2p link and a stub link with default or configured range

Hub and Spoke Example 1

6. Hub Router Stub Neighbor Support Discovery and misconfiguration detection

To avoid the possibility of any routing loops and misconfigurations due to partial deployments, this draft defines a new OSPF Router Functional Capability known as a Hub Router Stub Neighbor Support Capability. The value of this capability is a bit value to be assigned by IANA from OSPF Router Functional Capability Bits registry [I-D.ietf-ospf-rfc4970bis].

The Auto Discovery via announcement of the Hub Router Stub Neighbor Support Functional Capability ensures that the detection of Hub Routers configured with the feature and advertising both a normal and a modified router LSA to a spoke.

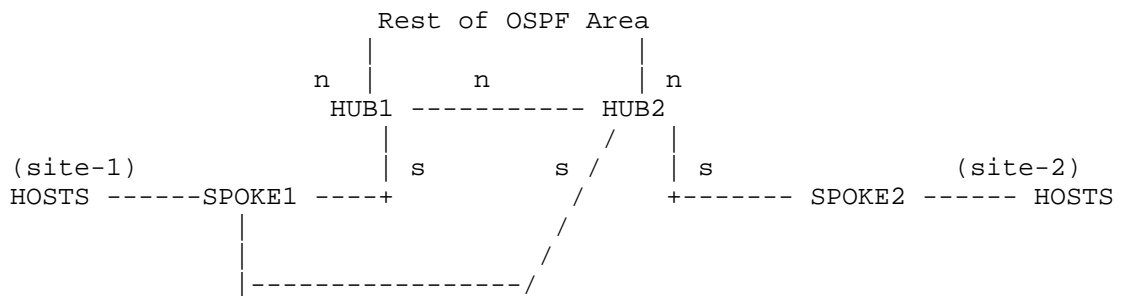
The deployment scenario assumes that all hubs will be upgraded with the new functionality and configure their link to their the spokes appropriately to prevent the modified router LSA of any hub to be flooded back over normal links.

A hub router receiving back its own modified local router LSA over one of its non-stub neighbors is an indication of misconfiguration and it SHOULD revert back to normal mode or log an error so the operation can intervene.

If a hub router receive both a normal router LSA over a normal link and a modified router LSA with aggregation of a known Hub Router with stub neighbor support over a stub link then

1. It should acknowledge the LSA to form the adjacency but not flood it over its normal links
2. It MUST ignore the Local Router LSA and use the normal router LSA in its own SPF calculations

Any hub router receiving back a modified local router LSA over one of its non-stub neighbors is an indication of misconfiguration and it SHOULD log an error so the operation can intervene.



(s) Stub neighbor links and (n) normal links.

Hub and Spoke Example 2

The dual homed spoke1 will flood the Local RTR LSAs to the hub. The hubs will not propagate the flooding of local rtr lsa on any normal links. If one of the hubs is misconfigured then the originator can detect the misconfiguration if its receives the local router lsa over a normal link.

Implementations are encouraged to provide a knob to manually override and enforcement the functionality in partial deployment scenarios for cases where the topology guarantees that the router supporting the stub neighbor will not cause routing loops.

7. Receiving and propagation of spoke routes

Hub router upon receiving the route from the spoke SHOULD NOT treat that route as an intra area route. For interoperability reason rest of the network does not have to have any knowledge of this new adjacency.

A hub router that acts as an ABR just converts the entire stub neighbor routes as if they were part of an area. Since in case of OSPF area, id is not carried and only the Hub router understands that it is connected to stub neighbor it can convert all the stub neighbor and treat them as part of single area. Since the hub router is filtering all the LSA it is well aware of all the neighbors being part of the same area.

Hub router will be able to summarize at the area boundary. That way all the spokes could be summarized into a single route.

8. Demand Circuit

Sections 4.1, 4.2 described how to reduce the amount of information flooded and increase scalability. The use of Demand Circuit capability can further enhance the scalability for some use cases.

By making the spoke neighbors as demand circuit we will be able to suppress the refresh of all the routes we have learned from spoke sites. Only incremental changes are flooded in the network. Most networks have large number of spoke sites, in some large network there could be around 18-20K spoke sites each sending up to 3-5 subnets. Have to refresh these large number of LSAs can have unnecessary information flooded throughout large OSPF domain.

Second type of spoke sites that are emerging are running over long distance wireless networks. Sending periodic hellos for neighbor detection is not desired behavior in long distance wireless network. We do understand this can have convergence impact for the spoke that is dual homed.

9. Benefits

By making hub router define a stub neighbor we would be able to run OSPF in a true hub and spoke setup. Where the router that connects to the network and has local routes that needs to be advertising to rest of the network does not have to participate in the larger OSPF topology. Also the core network does not get destabilize due to flaps on the spoke churns causing impact on core convergence.

10. Security Considerations

This memo does not introduce any new security concerns or take any directed action towards improving the security of OSPF deployments in general. However, since all links in between OSPF neighbors do not add to router link states it could be considered as a security improvement by protecting an adjacency that can have larger network impact.

11. IANA Considerations

There are no IANA considerations.

12. Acknowledgments

This document was produced using Marshall Rose's xml2rfc tool.

13. Normative References

- [I-D.ietf-ospf-rfc4970bis]
Lindem, A., Shen, N., Vasseur, J., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", draft-ietf-ospf-rfc4970bis-07 (work in progress), October 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <<http://www.rfc-editor.org/info/rfc2328>>.

Authors' Addresses

Khalid Raza
vIPtela
1735 Technology Drive
San Jose, CA 95110
USA

EMail: khalid.raza@viptela.com

John Cavanaugh
405Labs
6285 Lusk Blvd
San Diego, CA 92121
USA

EMail: John@405labs.com

Andrew Kulawiak
Morgan Stanley
1 New York Plaza
New York, NY 10004
USA

EMail: andrew.kulawiak@bankofamerica.com

Padma Pillay-Esnault
Cisco Systems
510 McCarty Blvd
Milpitas, CA 95035
USA

EMail: ppe@cisco.com

Faraz Shamim
Cisco Systems
2200 President George Bush TPKE
Richardson, TX 75082
USA

EMail: sshamim@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 30, 2015

L. Wang
S. Hares
N. Wu
Huawei
September 26, 2014

Yang Data model for I2RS interface to the OSPF protocol
draft-wang-i2rs-ospf-dm-00

Abstract

OSPF (OSPFv2 and OSPFv3) is widely deployed link-state protocol in routing networks. During the past decades, it has been operated and maintained through typical CLI, SNMP and NETCONF. With the expansion and complication of modern networks, the necessity for rapid and dynamic control has been increased. The I2RS is a standard-based interface which provides a programmatic way to achieve this goal.

This document specifies an OSPF yang data model for the I2RS interface to OSPF. This model is based on the the I2RS OSPF informational model (draft-ietf-wu-ospf-info-model-00) which satisfies the requirements suggested by the I2RS use case requirements for the IGP. This yang data model can be used by I2RS client-agent protocol to program OSPF routing entities.

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 <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 30, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Yang Tree Diagrams	3
2. OSPF data	3
3. I2RS OSPF Data Model	4
4. Relationship to other I2RS Data Models	15
5. OSPF Yang Data Model	15
6. IANA Considerations	52
7. Security Considerations	53
8. Acknowledgements	53
9. References	53
9.1. Informative References	53
9.2. Normative References	53
Authors' Addresses	54

1. Introduction

As one of well-known link-state protocols, OSPF[RFC2328] has been widely used in the routing of intra domain networks. During the past decades, it has been deployed with the help of typical interfaces such as CLI, SNMP and NETCONF. As modern networks grow in scale and complexity, the necessity for rapid and dynamic control has been increased. The I2RS[I-D.ietf-i2rs-architecture] is a standard-based interface which provides a programmatic way to achieve this goal.

This document specifies an yang data model for I2RS interface to the OSPF protocol based on the I2RS information model specified in draft-ietf-wu-ospf-info-model-00.

In order to support large intra-domain, OSPF has been organized hierarchically into areas. The topology of one area is hidden from the rest of networks, which is beneficial from the reduction of

routing traffic. Based on flooding mechanism, each routing-system in one OSPF area will maintain the identical database from which a pair-wise shortest tree is calculated in the distributed manner. As one client of RIB, OSPF SHOULD populate its routing information into RIB as stated in [I-D.ietf-i2rs-rib-info-model]

1.1. Yang Tree Diagrams

The Yang Tree diagrams used in this draft utilized a simple graphical representation of the data model. The meaning of the symbols are as follows:

- o Brackets "[" and "]" enclose list keys
- o Abbreviations before data node names: "rw" mean configuration (read-write) and "ro" state diagrams.
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stand for the contents of subtree that are not shown.

Future yang symbols may be added to indicate the object relationship, ephemeral state, and other I2RS specific relationships in yang 1.1

2. OSPF data

This section describes the data involved in the OSPF information model in detail. Please note OSPF in this document means both OSPFv2 and OSPFv3[RFC5340]protocol unless specified. OSPF data includes information related to OSPF instance, OSPF area, OSPF multi-topology, OSPF interfaces, OSPF adjacencies and OSPF routes. A high-level architecture of the OSPF contents is shown as below.

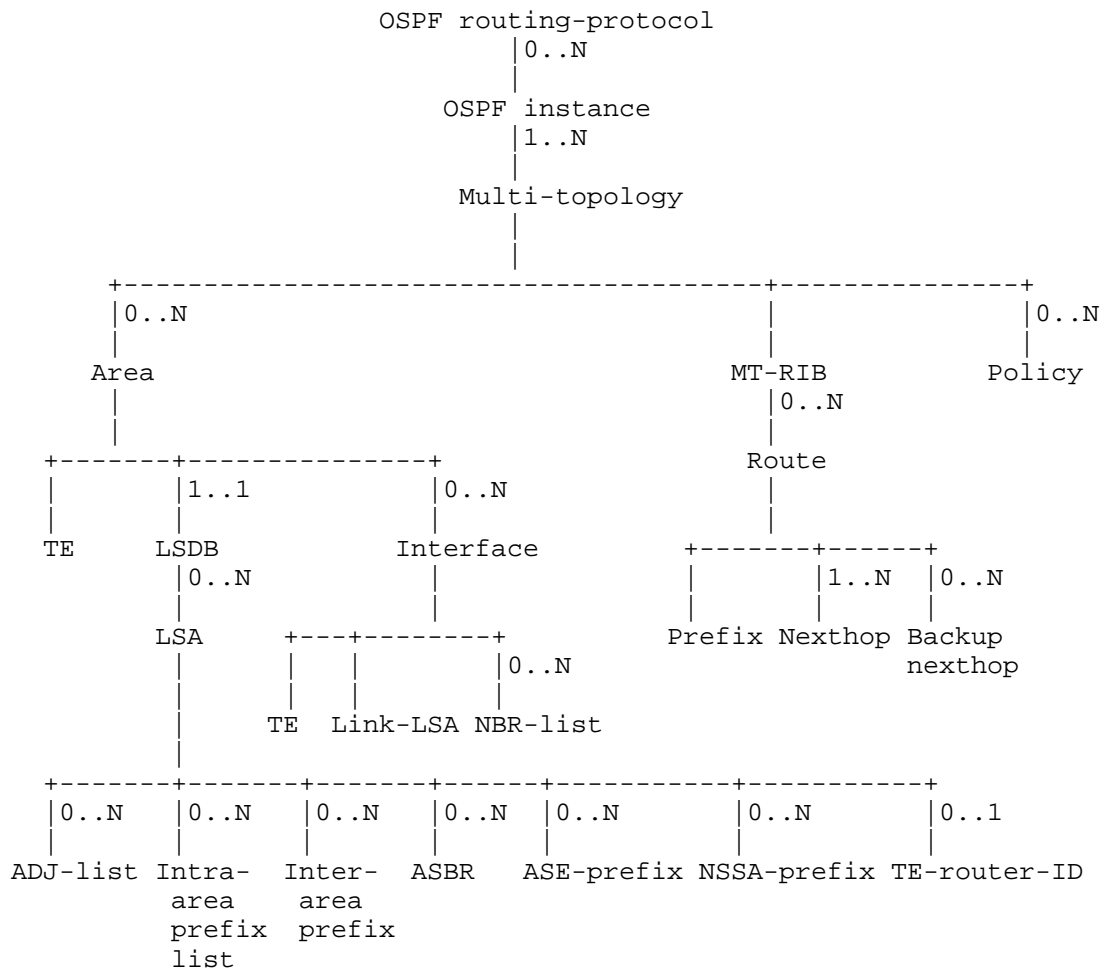


Figure 1: Architecture of OSPF information model

3. I2RS OSPF Data Model

```

    module: ospf-protocol
+--rw ospf-v4ur-instance
|   +--rw ospf-instance-name          string
|   +--rw ospf-vpn-name?              string
|   +--rw router-id                   inet:ip-address
|   +--ro protocol-status             protocol-status-def
|   +--ro ospf-type                   ospf-type-def
|   +--ro version                     ospf-version-def
|   +--ro ospf-process-create-mode    ospf-process-create-mode-def
|   +--rw preference                  uint32
|   +--rw hostname?                  string
|   +--rw mt-list
|       +--rw multi-topo* [mt-id]
|           +--rw mt-id                uint16
|           +--rw address-family      address-family-def
|           +--rw mt-status?          enumeration
|           +--rw policy-list* [policy-id]
|               +--rw policy-id      string
|           +--rw mt-rib
|               +--rw route* [prefix]
|                   +--rw prefix          inet:ipv4-prefix
|                   +--rw nexthop-list
|                       +--rw nexthop* [ospf-nexthop]
|                           +--rw ospf-nexthop    inet:ipv4-prefix
|                   +--rw back-nexthop?    inet:ipv4-prefix
|                   +--rw metric?          uint32
|                   +--rw type?           ospf-route-type-def
|                   +--rw route-state-info
|                       +--rw metric?          uint32
|                       +--rw route-current-state?    ospf-route-state-def
|                       +--rw route-previous-state?  ospf-route-state-def
|                       +--rw route-chg-reason?      route-chg-reason-def
|                       +--rw lsid?                inet:ip-address
|                       +--rw lsa-type?            lsa-type-def
|                       +--rw advertiser?          inet:ip-address

```



```

+--rw area-list
|   +--rw area-id                uint16
|   +--rw area-type?             area-type-def
|   +--rw area-status?           area-status-def
|   +--rw lsa-arrival-int?        uint32
|   +--rw lsa-orig-int?          uint32
|   +--rw router-number?         uint32
|   +--rw area-auth
|   |   +--rw (auth-mode-type)?
|   |   |   +--:(mode-simple)
|   |   |   |   +--rw simple-password?    string
|   |   |   +--:(mode-md5)
|   |   |   |   +--rw md5-password?       string
|   |   |   +--:(mode-hmac-sha256)
|   |   |   |   +--rw hmac-key-id?         uint32
|   |   |   |   +--rw hmac-password?      string
|   |   |   +--:(mode-keychain)
|   |   |   |   +--rw keychain-key-id?     uint32
|   |   |   |   +--rw keychain-password?  string
|   |   |   |   +--rw keychain-mode?      enumeration
|   |   |   |   +--rw keychain-periodic?  enumeration
|   |   |   |   +--rw send_time?          uint32
|   |   |   |   +--rw receive_tim?        uint32

```

```

+--rw lsdb
+--rw lsa*[lsa-v2-type link-state-id advertiser-id]
+--rw lsa-age?          uint32
+--rw lsa-options?      uint8
+--rw lsa-v2-type       enumeration
+--rw link-state-id     inet:ipv4-address
+--rw advertiser-id     inet:ip-prefix
+--rw seq-no?          uint32
+--rw chksum?          uint32
+--rw lsa-length?      uint32
+--rw (ls-type)?
+--:(ospf-v2-router-lsa)
+--rw ospf-v2-router-lsa
+--rw bit-flag         uint16
+--rw link-num         uint16
+--rw link-list* [link-id link-data]
+--rw link-id          inet:ipv4-address
+--rw link-data        inet:ipv4-address
+--rw link-type        enumeration
+--rw mt-num           uint16
+--rw metric           uint16
+--rw mt-metric* [mt-id]
+--rw mt-id            uint16
+--rw metric?         uint16
+--:(ospf-v2-network-lsa)
+--rw ospf-v2-network-lsa
+--rw network-mask     inet:ipv4-prefix
+--rw attached-router* [router-id]
+--rw router-id        inet:ipv4-address
+--:(ospf-v2-summary-lsa)
+--rw ospf-v2-summary-lsa
+--rw network-mask     inet:ipv4-prefix
+--rw mt-metric* [mt-id]
+--rw mt-id            uint16
+--rw metric?         uint16
+--:(ospf-v2-as-external-lsa)
+--rw ospf-v2-as-external-lsa
+--rw network-mask     inet:ipv4-prefix
+--rw mt-metric* [mt-id]
+--rw e-bit?          uint8
+--rw mt-id            uint8
+--rw metric?         uint16
+--rw forwarding-address?
+--rw                  inet:ipv4-address
+--rw external-route-tag? uint32
+--:(ospf-v2-nssa-external-lsa)
+--rw ospf-v2-nssa-external-lsa

```

```

|         +--rw network-mask      inet:ipv4-prefix
|         +--rw mt-metric* [mt-id]
|             +--rw e-bit?        uint8
|             +--rw mt-id         uint8
|             +--rw metric?       uint32
|             +--rw forwarding-address?
|                 inet:ipv4-address
|             +--rw external-route-tag? uint32
+---:(ospf-v2-te-router-lsa)
|   +--rw ospf-v2-te-router-lsa
|       +--rw type?              uint8
|       +--rw length?            uint32
|       +--rw router-id?         inet:ipv4-address
+---:(ospf-te-link-lsa)
|   +--rw ospf-te-link-lsa
|       +--rw type?              uint8
|       +--rw length?            uint32
|       +--rw link-type-stlv
|           +--rw type?          uint8
|           +--rw length?        uint32
|           +--rw link-type?     enumeration
+--rw link-id-tlv-stlv
|   +--rw type?                  uint8
|   +--rw length?                uint32
|   +--rw link-id?               inet:ipv4-address
+--rw local-address-stlv
|   +--rw type?                  uint8
|   +--rw length?                uint32
|   +--rw local-address-list*
|       [remote-address]
|       +--rw remote-address
|           inet:ipv4-address
+--rw remote-address-stlv
|   +--rw type?                  uint8
|   +--rw length?                uint32
|   +--rw remote-address-list*
|       [remote-address]
|       +--rw remote-address
|           inet:ipv4-address
+--rw te-metric-stlv
|   +--rw type?                  uint8
|   +--rw length?                uint32
|   +--rw value?                 uint32
+--rw maximum-bandwidth-stlv
|   +--rw type?                  uint8
|   +--rw length?                uint32
|   +--rw value?                 uint32
+--rw maximum-reservable-bandwidth-stlv

```

```
| | +--rw type? uint8  
| | +--rw length? uint32  
| | +--rw value? uint32  
+--rw unreserved-bandwidth-stlv  
| | +--rw type? uint8  
| | +--rw length? uint32  
| | +--rw value? uint32  
+--rw administrative-group-stlv  
| | +--rw type? uint8  
| | +--rw length? uint32  
| | +--rw value? uint32  
  
+--rw interface-list  
+--rw interface* [interface-index]  
|   +--rw interface-index uint64  
|   +--rw interface-name? string  
|   +--rw interface-status? interface-status-def  
|   +--rw interface-down-reason?  
|       interface-down-reason-def  
+--rw interface-net-type? interface-net-type-def  
+--rw interface-role? interface-role-def  
+--rw interface-te-info  
|   +--rw admin_group? uint32  
|   +--rw max_bandwidth? uint32  
|   +--rw max_rsv_bandwidth? uint32  
|   +--rw unrsv_bandwidth? uint32  
+--rw interface-auth  
|   +--rw (auth-mode-type)?  
|       |--:(mode-simple)  
|           |   +--rw simple-password? string  
|               |--:(mode-md5)  
|                   |   +--rw md5-password? string  
|                       |--:(mode-hmac-sha256)  
|                           |   +--rw hmac-key-id? uint32  
|                               |   +--rw hmac-password? string  
|                                   |--:(mode-keychain)  
|                                       |   +--rw keychain-key-id? uint32  
|                                           |   +--rw keychain-password? string  
|                                               |   +--rw keychain-mode? enumeration  
|                                                   |   +--rw keychain-periodic? enumeration  
|                                                       +--rw send_time? uint32  
|                                                           +--rw receive_tim? uint32  
+--rw ip-address? inet:ipv4-address  
+--rw nbr-list  
    +--rw nbr* [router-id]  
        +--rw router-id inet:ip-address  
        +--rw interface-index? uint64  
        +--rw interface-name? string
```

```

|                                     |--rw nbr-status?          nbr-status-def
|                                     |--rw nbr-previous-status? nbr-status-def
|                                     |--rw nbr-down-reason?    nbr-down-reason-def
|                                     |--rw nbr-address?       inet:ipv4-address
|                                     |--rw ip-address?        inet:ipv4-address
|--rw network-list* [network-prefix mask]
|   |--rw network-prefix    inet:ipv4-prefix
|   |--rw mask              inet:ipv4-prefix
|--rw route-info-list* [route-info-index]
|   |--rw route-info-index  uint32
|   |--rw router-id         inet:ipv4-address
|   |--rw ip-address-list* [ip-address]
|       |--rw ip-address    inet:ipv4-address

+--rw ospf-v6ur-instance
+--rw ospf-instance-name      string
+--rw ospf-vpn-name?         string
+--rw router-id              inet:ip-address
+--ro protocol-status        protocol-status-def
+--ro ospf-type              ospf-type-def
+--ro version                ospf-version-def
+--ro ospf-process-create-mode ospf-process-create-mode-def
+--rw preference             uint32
+--rw hostname?             string
+--rw mt-list
+--rw multi-topo* [mt-id]
+--rw mt-id                  uint16
+--rw address-family         address-family-def
+--rw mt-status?            enumeration
+--rw policy-list* [policy-id]
|   |--rw policy-id         string
+--rw mt-rib
|   |--rw route* [prefix]
|       |--rw prefix          inet:ipv6-prefix
|       |--rw nexthop-list
|           |--rw nexthop* [ospf-nexthop]
|               |--rw ospf-nexthop inet:ipv6-prefix
|       |--rw back-nexthop?    inet:ipv6-prefix
|       |--rw metric?         uint32
|       |--rw type?           ospf-route-type-def
|       |--rw route-state-info
|           |--rw metric?      uint32
|           |--rw route-current-state? ospf-route-state-def
|           |--rw route-previous-state? ospf-route-state-def
|           |--rw route-chg-reason?    route-chg-reason-def
|           |--rw lsid?            inet:ip-address
|           |--rw lsa-type?        lsa-type-def
|           |--rw advertiser?      inet:ip-address

```

```

+--rw area-list
  +--rw area* [area-id]
    +--rw area-id          uint16
    +--rw area-type?       area-type-def
    +--rw area-status?     area-status-def
    +--rw lsa-arrival-int?  uint32
    +--rw lsa-orig-int?    uint32
    +--rw router-number?   uint32
    +--rw area-auth
      +--rw (auth-mode-type)?
        +--:(mode-simple)
          | +--rw simple-password?    string
        +--:(mode-md5)
          | +--rw md5-password?       string
        +--:(mode-hmac-sha256)
          | +--rw hmac-key-id?        uint32
          | +--rw hmac-password?      string
        +--:(mode-keychain)
          +--rw keychain-key-id?      uint32
          +--rw keychain-password?    string
          +--rw keychain-mode?        enumeration
          +--rw keychain-periodic?    enumeration
          +--rw send_time?            uint32
          +--rw receive_tim?          uint32
    +--rw lsdb
      +--rw lsa* [lsa-v3-type link-state-id advertiser-id]
        +--rw lsa-age?          uint32
        +--rw lsa-v3-type       enumeration
        +--rw link-state-id     uint32
        +--rw advertiser-id     inet:ip-prefix
        +--rw seq-no?           uint32
        +--rw chksum?           uint32
        +--rw lsa-length?       uint32
        +--rw (ls-type)?
          +--:(ospf-v3-router-lsa)
            +--rw ospf-v3-router-lsa
              +--rw option          uint16
              +--rw link-list*
                [link-type interface-id neighbor-interface-id]
                  +--rw link-type    enumeration
                  +--rw metric?      uint32
                  +--rw interface-id uint32
                  +--rw neighbor-interface-id uint32
                  +--rw neighbor-router-id?
                    inet:ipv4-address
          +--:(ospf-v3-network-lsa)
            +--rw ospf-v3-network-lsa
              +--rw option          uint32

```

```

|         +--rw link-list* [attached-router-id]
|         |         +--rw attached-router-id
|         |         |         inet:ipv4-address
+---:(ospf-v3-inter-area-prefix-lsa)
|         +--rw ospf-v3-inter-area-prefix-lsa
|         |         +--rw metric?                uint32
|         |         +--rw prefix-length           uint8
|         |         +--rw prefix-options          uint8
|         |         +--rw address-prefix-list* [address-prefix]
|         |         |         +--rw address-prefix  inet:ipv6-prefix
+---:(ospf-v3-inter-area-router-lsa)
|         +--rw ospf-v3-inter-area-router-lsa
|         |         +--rw options                  uint8
|         |         +--rw metric?                  uint32
|         |         +--rw destination-router-id?
|         |         |         inet:ipv4-address
+---:(ospf-v3-as-external-lsa)
|         +--rw ospf-v3-as-external-lsa
|         |         +--rw options                  uint16
|         |         +--rw metric                  uint16
|         |         +--rw prefix-length           uint8
|         |         +--rw prefix-options          uint8
|         |         +--rw referenced-ls-type       uint8
|         |         +--rw address-prefix-list* [address-prefix]
|         |         |         +--rw address-prefix  inet:ipv6-prefix
|         |         +--rw forwarding-address?     inet:ipv6-prefix
|         |         +--rw external-route-tag?     uint32
|         |         +--rw referenced-link-state-id? uint32
+---:(ospf-v3-nssa-lsa)
|         +--rw ospf-v3-nssa-lsa
|         |         +--rw options                  uint16
|         |         +--rw metric                  uint16
|         |         +--rw prefixlength            uint8
|         |         +--rw prefixoptions           uint8
|         |         +--rw referenced-ls-type       uint8
|         |         +--rw address-prefix-list* [address-prefix]
|         |         |         +--rw address-prefix  inet:ipv6-prefix
|         |         +--rw forwarding-address?     inet:ipv6-prefix
|         |         +--rw external-route-tag?     uint32
|         |         +--rw referenced-link-state-id? uint32
+---:(ospf-v3-link-lsa)
|         +--rw ospf-v3-link-lsa
|         |         +--rw priority                uint8
|         |         +--rw options                  uint32
|         |         +--rw link-local-interface-address?
|         |         |         inet:ipv6-address
|         |         +--rw prefixes                uint32
|         |         +--rw address-prefix-list*

```

```

|                                     [address-prefix-index]
|         +--rw address-prefix-index      uint32
|         +--rw prefix-length             uint8
|         +--rw prefix-options?           uint8
|         +--rw address-prefix* [address]
|             +--rw address      inet:ipv6-prefix
+---:(ospf-v3-intra-area-prefix-lsa)
|   +--rw ospf-v3-intra-area-prefix-lsa
|       +--rw prefixes                uint32
|       +--rw referenced-ls-type       uint16
|       +--rw referenced-link-state-id uint32
|       +--rw referenced-advertising-router
|           inet:ipv4-address
|       +--rw address-prefix-list*
|           [address-prefix-index]
|               +--rw address-prefix-index      uint32
|               +--rw prefix-length             uint8
|               +--rw prefix-options            uint8
|               +--rw address-prefix* [address]
|                   +--rw address      inet:ipv6-prefix
+---:(ospf-v3-te-router-ipv6-address-lsa)
|   +--rw ospf-v3-te-router-ipv6-address
|       +--rw type          uint8
|       +--rw length        uint16
|       +--rw router-id     inet:ipv6-address
+---:(te-link-lsa)
|   +--rw ospf-te-link-lsa
|       +--rw type?          uint8
|       +--rw length?        uint32
|       +--rw link-type-stlv
|           +--rw type?          uint8
|           +--rw length?        uint32
|           +--rw link-type?     enumeration
+--rw link-id-tlv-stlv
|   +--rw type?          uint8
|   +--rw length?        uint32
|   +--rw link-id?      inet:ipv4-address
+--rw local-address-stlv
|   +--rw type?          uint8
|   +--rw length?        uint32
|   +--rw local-address-list*
|       [remote-address]
|       +--rw remote-address
|           inet:ipv4-address
+--rw remote-address-stlv
|   +--rw type?          uint8
|   +--rw length?        uint32
|   +--rw remote-address-list*

```



```

|                                     [remote-address]
|                                     +--rw remote-address
|                                     |   inet:ipv4-address
+--rw te-metric-stlv
|   +--rw type?      uint8
|   +--rw length?    uint32
|   +--rw value?     uint32
+--rw maximum-bandwidth-stlv
|   +--rw type?      uint8
|   +--rw length?    uint32
|   +--rw value?     uint32
+--rw maximum-reservable-bandwidth-stlv
|   +--rw type?      uint8
|   +--rw length?    uint32
|   +--rw value?     uint32
+--rw unreserved-bandwidth-stlv
|   +--rw type?      uint8
|   +--rw length?    uint32
|   +--rw value?     uint32
+--rw administrative-group-stlv
|   +--rw type?      uint8
|   +--rw length?    uint32
|   +--rw value?     uint32
+--rw interface-list
|   +--rw interface* [interface-index]
|       +--rw interface-index      uint64
|       +--rw interface-name?      string
|       +--rw interface-status?    interface-status-def
|       +--rw interface-down-reason?
|           interface-down-reason-def
+--rw interface-net-type?    interface-net-type-def
+--rw interface-role?        interface-role-def
+--rw interface-te-info
|   +--rw admin_group?         uint32
|   +--rw max_bandwidth?       uint32
|   +--rw max_rsv_bandwidth?   uint32
|   +--rw unrsv_bandwidth?     uint32
+--rw interface-auth
|   +--rw (auth-mode-type)?
|       +--:(mode-simple)
|           +--rw simple-password?    string
|       +--:(mode-md5)
|           +--rw md5-password?       string
|       +--:(mode-hmac-sha256)
|           +--rw hmac-key-id?         uint32
|           +--rw hmac-password?      string
|       +--:(mode-keychain)
|           +--rw keychain-key-id?     uint32

```

```

|         |--rw keychain-password?    string
|         |--rw keychain-mode?        enumeration
|         |--rw keychain-periodic?    enumeration
|         |--rw send_time?            uint32
|         |--rw receive_tim?          uint32
|--rw ip-address                      inet:ipv6-address
|--rw nbr-list
|   |--rw nbr* [router-id]
|   |--rw router-id                  inet:ip-address
|   |--rw interface-index?          uint64
|   |--rw interface-name?          string
|   |--rw nbr-status?               nbr-status-def
|   |--rw nbr-previous-status?      nbr-status-def
|   |--rw nbr-down-reason?          nbr-down-reason-def
|   |--rw nbr-address?              inet:ipv6-address
|   |--rw ip-address                 inet:ipv6-address
|--rw network-list* [network-index]
|   |--rw network-index             uint32
|   |--rw network-prefix            inet:ipv4-prefix
|   |--rw mask                      inet:ipv4-prefix
|--rw route-info-list* [route-info-index]
|   |--rw route-info-index          uint32
|   |--rw router-id                 inet:ipv4-address
|   |--rw ip-address-list* [ip-address]
|   |--rw ip-address                inet:ipv4-address

```

Figure 2 top-level I2RS YANG model of OSPF

4. Relationship to other I2RS Data Models

(TBD)

5. OSPF Yang Data Model

```

module ospf-protocol {

  namespace "urn:huawei:params:xml:ns:yang:rt:i2rs:i2rs-ospf";
  // replace with iana namespace when assigned
  prefix "i2rs-ospf";

  import ietf-inet-types {
    prefix inet;
    //rfc6991
  }

  organization "Huawei Technologies Co., Ltd.";
  contact
    "Email: wanglixing@huawei.com

```

```
Email: shares@ndzh.com
Email: eric.wu@huawei.com";

revision "2014-08-22" {
  description "initial revision";
  reference "draft-wu-i2rs-ospf-info-model-00";
}

typedef address-family-def {
  description
    "tbd.";
  type enumeration {
    enum "v4ur";
    enum "v6ur";
    enum "v4mr";
    enum "v6mr";
  }
}

typedef ospf-type-def {
  type enumeration {
    enum "asbr";
    enum "abr";
  }
}

typedef ospf-route-type-def {
  description
    "The type of ospf route.";
  type enumeration {
    enum "ospf type 1";
    enum "ospf type 2";
    enum "ospf type 3";
    enum "ospf type 4";
    enum "ospf type 5";
    enum "ospf type 7";
  }
}

typedef lsa-type-def {
  description
    "The type of ospf lsa.";
  type enumeration {
    enum "route lsa";
    enum "network lsa";
    enum "summary3 lsa";
    enum "summary4 lsa";
    enum "ase lsa";
  }
}
```

```
        enum "nssa lsa";
        enum "intter-area-prefix lsa";
        enum "inter-area-router lsa";
        enum "link lsa";
        enum "intra-area-prefix lsa";
        enum "te router-id lsa";
        enum "link-te lsa";
    }
}

typedef ospf-route-state-def {
    type enumeration {
        enum "active";
        enum "inactive";
        enum "primary";
        enum "backup";
    }
}

typedef route-chg-reason-def {
    description
        "The changing reason of ospf route .";
    type enumeration {
        enum "orig-adv";
        enum "orig-withdraw";
        enum "adj-down";
        enum "policy-deny";
    }
}

typedef area-status-def {
    type enumeration {
        enum "active";
        enum "reset";
        enum "shutdown";
    }
}

typedef area-type-def {
    type enumeration {
        enum "normal";
        enum "stub";
        enum "nssa";
    }
}

typedef lsdb-status-def {
    type enumeration {
```

```
        enum "normal";
        enum "overflow";
    }
}

typedef interface-net-type-def {
    type enumeration {
        enum "p2p";
        enum "broadcast";
        enum "nbma";
        enum "p2mp";
    }
}

typedef interface-status-def {
    type enumeration {
        enum "if-up";
        enum "if-down";
    }
}

typedef interface-down-reason-def {
    type enumeration {
        enum "phy-down";
        enum "admin-down";
        enum "ip-down";
        enum "i2rs-down";
    }
}

typedef nbr-status-def {
    type enumeration {
        enum "down";
        enum "attempt";
        enum "2-way";
        enum "exstat";
        enum "exchange";
        enum "loading";
        enum "full";
    }
}

typedef nbr-down-reason-def {
    type enumeration {
        enum "if-down";
        enum "bfd-down";
        enum "expiration";
        enum "cfd-chg";
        enum "i2rs-down";
    }
}
```

```
    }  
  }  
  typedef interface-role-def {  
    type enumeration {  
      enum "dr";  
      enum "bdr";  
    }  
  }  
  
  typedef protocol-status-def {  
    type enumeration {  
      enum "active";  
      enum "reset";  
      enum "shutdown";  
      enum "overload";  
    }  
  }  
  
  typedef ospf-version-def {  
    description  
      "OSPF v2 is for IPV4, and ospf v3 is for IPV6.";  
    type enumeration {  
      enum "v2";  
      enum "v3";  
    }  
  }  
  
  typedef ospf-process-create-mode-def {  
    type enumeration {  
      enum "not-i2rs";  
      enum "i2rsclient-create-ospf-instance";  
      enum "i2rsagent-fails-ospf-instance-create";  
      enum "i2rsagent-created-ospf-instance";  
      enum "i2rsagent-ospf-instance-create";  
      enum "i2rsagent-rejects-ospf-instance-create";  
      enum "i2rsagent-attempts-ospf-instance-create";  
    }  
  }  
  
  grouping ospf-instance-commom {  
    description  
      "the common structure of ospf process.";  
    leaf ospf-instance-name {  
      type string;  
      mandatory true;  
    }  
  }
```

```
leaf ospf-vpn-name {
    type string;
    mandatory false;
}

leaf router-id {
    type inet:ip-address;
    mandatory true;
}

leaf protocol-status {
    type protocol-status-def;
    config "false";
    mandatory true;
}

leaf ospf-type {
    type ospf-type-def;
    config "false";
    mandatory true;
}

leaf version {
    type ospf-version-def;
    config "false";
    mandatory true;
}

leaf ospf-process-create-mode {
    type ospf-process-create-mode-def;
    config "false";
    mandatory true;
}

leaf preference {
    type uint32 {
        range "1..4294967295";
    }
    mandatory true;
}

leaf hostname {
    type string;
    mandatory false;
}
}
```

```
grouping ospf-mt-commom {
  description
    "the common structure of ospf process.";
  leaf mt-id {
    type uint16;
  }
  leaf address-family {
    type address-family-def;
    mandatory true;
  }

  leaf mt-status {
    type enumeration {
      enum "active";
      enum "inactive";
    }
  }

  list policy-list {
    description
      "The policy of this MT.";
    key "policy-id";
    leaf policy-id {
      type string;
    }
  }
}

grouping auth-info {
  choice auth-mode-type {
    case mode-simple {
      leaf simple-password {
        type string;
      }
    }
    case mode-md5 {
      leaf md5-password {
        type string;
      }
    }
    case mode-hmac-sha256 {
      leaf hmac-key-id {
        type uint32;
      }
      leaf hmac-password {
        type string;
      }
    }
  }
}
```



```
    case mode-keychain {
      leaf keychain-key-id {
        type uint32;
      }
      leaf keychain-password {
        type string;
      }
      leaf keychain-mode {
        type enumeration {
          enum "absolute";
          enum "periodic";
        }
      }
    }

    leaf keychain-periodic {
      type enumeration {
        enum "daily";
        enum "weekly";
        enum "monthly";
        enum "yearly";
      }
    }
    leaf send_time {
      type uint32;
    }
    leaf receive_tim {
      type uint32;
    }
  }
}

grouping ospf-area-commom {
  description
    "the area structure of ospf process.";
  leaf area-id {
    description "Tbd.";
    type uint16;
  }

  leaf area-type {
    type area-type-def;
  }

  leaf area-status {
    type area-status-def;
  }
}
```

```
    }

    leaf lsa-arrival-int {
        type uint32;
    }
    leaf lsa-orig-int {
        type uint32;
    }
    leaf router-number {
        type uint32;
    }
    container area-auth{
        uses auth-info;
    }
}

grouping ospf-route-common {
    description
        "the common structure of ospf route.";
    leaf metric {
        type uint32;
    }

    leaf type {
        type ospf-route-type-def;
    }

    container route-state-info {
        leaf metric {
            type uint32;
        }

        leaf route-current-state {
            type ospf-route-state-def;
        }

        leaf route-previous-state {
            type ospf-route-state-def;
        }

        leaf route-chg-reason {
            type route-chg-reason-def;
        }

        leaf lsid {
            type inet:ip-address;
        }
    }
}
```

```
    leaf lsa-type {
      type lsa-type-def;
    }

    leaf advertiser {
      type inet:ip-address;
    }
  }
}

grouping ospf-interface-common {
  description
    "the area structure of ospf interface.";
  leaf interface-index {
    description "Tbd.";
    type uint64;
  }

  leaf interface-name {
    description "Tbd.";
    type string;
  }

  leaf interface-status {
    type interface-status-def;
  }

  leaf interface-down-reason {
    type interface-down-reason-def;
  }

  leaf interface-net-type {
    type interface-net-type-def;
  }

  leaf interface-role {
    type interface-role-def;
  }
  container interface-te-info {
    leaf admin_group {
      type uint32;
    }
    leaf max_bandwidth {
      type uint32;
    }
    leaf max_rsv_bandwidth {
      type uint32;
    }
    leaf unrsv_bandwidth {
```

```
        type uint32;
    }
}
container interface-auth{
    uses auth-info;
}
}

grouping ospf-nbr-commom {
    description
        "the area structure of ospf nbr.";
    leaf router-id {
        type inet:ip-address;
    }

    leaf interface-index {
        description "Tbd.";
        type uint64;
    }

    leaf interface-name {
        description "Tbd.";
        type string;
    }

    leaf nbr-status {
        type nbr-status-def;
    }
    leaf nbr-previous-status {
        type nbr-status-def;
    }
    leaf nbr-down-reason {
        type nbr-down-reason-def;
    }
}

grouping ospf-v2-lsa-header-commom {
    description
        "the ospf v2 lsa header ";
    leaf lsa-age {
        type uint32;
    }
    leaf lsa-options {
        type uint8;
    }
    leaf lsa-v2-type {
        mandatory "true";
        type enumeration {
```

```
        enum router-lsa {
            value "1";
        }
        enum network-lsa {
            value "2";
        }
        enum summary-abr-lsa {
            value "3";
        }
        enum summary-asbr-lsa {
            value "4";
        }
        enum ase-lsa {
            value "5";
        }
        enum nssa-lsa {
            value "7";
        }
        enum te-lsa {
            description "export-extcommunity and import-extcommunity:";
            value "10";
        }
    }
}
leaf link-state-id {
    type inet:ipv4-address;
    mandatory true;
}
leaf advertiser-id {
    type inet:ip-prefix;
    mandatory true;
}
leaf seq-no {
    type uint32;
}

leaf chksum {
    type uint32;
}
leaf lsa-length {
    type uint32;
}
}

grouping ospf-v3-lsa-header-commom {
    description
        "the ospf v3 lsa header ";
    leaf lsa-age {
```

```
        type uint32;
    }
    leaf lsa-v3-type {
        mandatory "true";
        type enumeration {
            enum router-lsa {
                value "2001";
            }
            enum network-lsa {
                value "2002";
            }
            enum inter-area-prefix-lsa {
                value "2003";
            }
            enum inter-area-router-lsa {
                value "2004";
            }
            enum as-external-lsas {
                value "4005";
            }
            enum nssa-lsa {
                value "2007";
            }
            enum link-lsa {
                value "0008";
            }
            enum intra-area-prefix-lsa {
                value "2009";
            }
            enum te-lsa {
                value "10";
                description "Te:";
            }
        }
    }
}

leaf link-state-id {
    description "lsa type/scope unique identifier.";
    type uint32;
}
leaf advertiser-id {
    type inet:ip-prefix;
    mandatory true;
}
leaf seq-no {
    type uint32;
}
```

```
    leaf chksum {
      type uint32;
    }
    leaf lsa-length {
      type uint32;
    }
  }
  grouping ospf-v2-router-lsa {
    container ospf-v2-router-lsa {
      leaf bit-flag {
        description "bit V:When set, the router is
          an endpoint of one or more fully
          adjacent virtual links having the
          described area as Transit area
          (V is for virtual link endpoint).
          bit E:When set, the router is an AS boundary
          router (E is for external).
          bit B:When set, the router is an area
          border router (B is for border).";
        type uint16;
        mandatory true;
      }
      leaf link-num {
        description "The number of router links
          described in this LSA. This must be
          the total collection of router links
          (i.e., interfaces) to the area.";
        type uint16;
        mandatory true;
      }
    }
    list link-list{
      key "link-id link-data";
      leaf link-id {
        description "Identifies the object
          that this router link connects to. Value
          depends on the link's Type. When
          connecting to an object that also
          originates an LSA (i.e., another router
          or a transit network) the Link ID is equal
          to the neighboring LSA's Link
          State ID. This provides the key
          for looking up the neighboring
          LSA in the link state database
          during the routing table calculation.";
        type inet:ipv4-address;
        mandatory true;
      }
    }
  }
}
```

```

    leaf link-data{
        type inet:ipv4-address;
    }

    leaf link-type {
        type enumeration {
            enum "p2p";
enum "transit";
            enum "stub";
enum "virtual";
        }
        mandatory true;
    }
    leaf mt-num {
        type uint16;
        mandatory true;
    }
    leaf metric {
        type uint16;
        mandatory true;
    }
    list mt-metric{
        key "mt-id";
        leaf mt-id {
            type uint16;
        }
        leaf metric {
            type uint16;
        }
    }
}
}
}

grouping ospf-v2-network-lsa {
    container ospf-v2-network-lsa {
        leaf network-mask {
            description "The ip address mask for the
network.  for example, a class a
network would have the mask 0xff000000.";
            type inet:ipv4-prefix;
            mandatory true;
        }
        list attached-router{
            description "The router ids of each of the
routers attached to the network.
actually, only those routers that are fully
adjacent to the designated router are listed."

```



```

        the designated router includes itself in this list. ";
    key "router-id";
    leaf router-id {
        type inet:ipv4-address;
    }
}
}
}

grouping ospf-v2-summary-lsa {
    container ospf-v2-summary-lsa {
        leaf network-mask {
            description "for type 3 summary-lsas, this
            indicates the destination network's ip address
            mask.  for example, when advertising the
            location of a class a network the value 0xff000000 would be
            used.  this field is not meaningful and must be
            zero for type 4 summary-lsas.";
            type inet:ipv4-prefix;
            mandatory true;
        }

        list mt-metric{
            key "mt-id";
            leaf mt-id {
                type uint16;
            }
            leaf metric {
                type uint16;
            }
        }
    }
}

grouping ospf-v2-as-external-lsa {
    container ospf-v2-as-external-lsa {
        leaf network-mask {
            description "The ip address mask for the
            advertised destination.  for example,
            when advertising a class a network the
            mask 0xff000000 would be used.";
            type inet:ipv4-prefix;
            mandatory true;
        }

        list mt-metric{
            key "mt-id";
            leaf e-bit {

```

```

        description "The type of external metric.
        if bit e is set, the metric specified is a type
        2 external metric.  this means the metric is
        considered larger than any link state path.
        if bit e is zero, the specified metric is a
        type 1 external metric.  this means
        that it is expressed in the same units as
        the link state metric
        (i.e., the same units as interface cost)..";
        type uint8;
    }
    leaf mt-id {
        type uint8;
    }
    leaf metric {
        type uint16;
    }
    leaf forwarding-address {
        description "data traffic for the advertised
        destination will be forwarded to this address.
        if the forwarding address is set to 0.0.0.0,
        data traffic will be forwarded instead to the
        lsa's originator (i.e., the responsible as
        boundary router).";
        type inet:ipv4-address;
    }
    leaf external-route-tag {
        description "a 32-bit field attached to each external
        route.  this is not used by the ospf protocol itself.
        it may be used to communicate information between as
        boundary routers; the precise nature of
        such information is outside the scope of
        this specification.";
        type uint32;
    }
}
}
}

grouping ospf-v2-nssa-external-lsa {
    container ospf-v2-nssa-external-lsa {
        leaf network-mask {
            description "The ip address mask for the
            advertised destination.  for
            example, when advertising a class a
            network the mask 0xff000000
            would be used.";
            type inet:ipv4-prefix;
        }
    }
}

```

```
    mandatory true;
  }

  list mt-metric{
    key "mt-id";
    leaf e-bit {
      description "The type of external metric.
        if bit e is set, the metric specified is a
        type 2 external metric. this means the metric is
        considered larger than any link state path.
        If bit e is zero, the specified metric is a
        type 1 external metric. This means
        that it is expressed in the same units as
        the link state metric
        (i.e., the same units as interface cost)..";
      type uint8;
    }
    leaf mt-id {
      type uint8;
    }
    leaf metric {
      type uint32;
    }
    leaf forwarding-address {
      description "data traffic for the advertised
        destination will be forwarded to
        this address. if the forwarding address is
        set to 0.0.0.0, data traffic will be forwarded
        instead to the lsa's originator (i.e.,
        the responsible as boundary router).";
      type inet:ipv4-address;
    }
    leaf external-route-tag {
      description "a 32-bit field attached to each
        external route. this is not used by the ospf
        protocol itself. it may be used to communicate
        information between as boundary routers;
        the precise nature of such information is outside
        the scope of this specification.";
      type uint32;
    }
  }
}

grouping ospf-v2-te-router-lsa {
  container ospf-v2-te-router-lsa {
    description "The router address tlv specifies a
```

```
stable ip address of the advertising router that
is always reachable if there is any
connectivity to it; this is typically implemented
as a loopback address. the key attribute is that
the address does not become unusable if an interface
is down. in other protocols, this is known
as the router id, but for obvious reasons this
nomenclature is avoided here. if a router advertises
bgp routes with the bgp next hop attribute set to the
bgp router id, then the router address
should be the same as the bgp router id. ";
leaf type {
  description "The router address tlv is type 1,
    has a length of 4.";
  type uint8;
}
leaf length {
  description "The router address tlv has a length of 4.";
  type uint32;
}
leaf router-id {
  description "The value of router address tlv is the
    four octet ip address..";
  type inet:ipv4-address;
}
}
}

grouping ospf-te-link-lsa {
  container ospf-te-link-lsa {
    description "The link tlv describes a single link.
      It is constructed of a set of sub-tlvs. There are no
      ordering requirements for the sub-tlvs.";
    leaf type {
      description "The link tlv is type 2.";
      type uint8;
    }
    leaf length {
      description "The length of the link tlv is variable.";
      type uint32;
    }
    container link-type-stlv {
      description "The link type sub-tlv defines the
        type of the link.";
      leaf type {
        description "The link type sub-tlv is tlv type 1.";
        type uint8;
      }
    }
  }
}
```

```
    leaf length {
      description "The link type sub-tlv is one octet in length.";
      type uint32;
    }
    leaf link-type {
      description ".          1 - point-to-point      2 - multi-access.";
      type enumeration {
        enum "point-to-point";
        enum "multi-access";
      }
    }
  }
}

container link-id-tlv-stlv {
  description "The link id sub-tlv identifies the
    other end of the link. The link id is identical to the
    contents of the link id field in the
    router lsa for these link types.";
  leaf type {
    description "The link type sub-tlv is tlv type 2.";
    type uint8;
  }
  leaf length {
    description "The link type sub-tlv is four octet in length.";
    type uint32;
  }
  leaf link-id {
    description ".";
    type inet:ipv4-address;
  }
}

container local-address-stlv {
  description "The local interface ip address sub-tlv
    specifies the ip address(es) of the interface corresponding
    to this link. If there are multiple local addresses on
    the link, they are all listed in this sub-tlv.";
  leaf type {
    description "The local interface ip address sub-tlv is tlv type 3.";
    type uint8;
  }
  leaf length {
    description "The local interface ip address sub-tlv is 4n
      octets in length, where n is the number of neighbor addresses.";
    type uint32;
  }
  list local-address-list {
    key "remote-address";
  }
}
```

```
    leaf remote-address {
      type inet:ipv4-address;
    }
  }
}

container remote-address-stlv {
  description "The remote interface ip address sub-tlv
    specifies the ip address(es) of the neighbor's interface
    corresponding to this link. This and the
    local address are used to discern multiple parallel
    links between systems. If the link type of the link
    is multi-access, the remote interface ip address is
    set to 0.0.0.0; alternatively, an
    implementation may choose not to send this sub-tlv.";
  leaf type {
    description "The remote interface ip address sub-tlv is tlv type 4.";
    type uint8;
  }
  leaf length {
    description "The remote interface ip address sub-tlv is 4n
      octets in length, where n is the number of neighbor addresses.";
    type uint32;
  }
  list remote-address-list {
    key "remote-address";
    leaf remote-address {
      type inet:ipv4-address;
    }
  }
}

container te-metric-stlv {
  description "The traffic engineering metric sub-tlv
    specifies the link metric for traffic engineering purposes.
    This metric may be different than the
    standard ospf link metric. Typically, this metric
    is assigned by a network administrator..";
  leaf type {
    description "The traffic engineering metric
      sub-tlv is tlv type 5.";
    type uint8;
  }
  leaf length {
    description "The traffic engineering metric sub-tlv is
      four octets in length..";
    type uint32;
  }
}
```

```
    leaf value {
      type uint32;
    }
  }

  container maximum-bandwidth-stlv {
    description "The maximum bandwidth sub-tlv specifies
      the maximum bandwidth that can be used on this link,
      in this direction (from the system originating the lsa
      to its neighbor), in ieee floating point format.
      This is the true link capacity. The units are bytes
      per second. The maximum bandwidth sub-tlv is tlv type 6,
      and is four octets in length.";
    leaf type {
      description "The maximum bandwidth sub-tlv is tlv type 6.";
      type uint8;
    }
    leaf length {
      description "The maximum bandwidth sub-tlv is
        four octets in length.";
      type uint32;
    }
    leaf value {
      type uint32;
    }
  }

  container maximum-reservable-bandwidth-stlv {
    description "The maximum reservable bandwidth
      sub-tlv specifies the maximum bandwidth that may
      be reserved on this link, in this direction, in
      ieee floating point format. note that this may be
      greater than the maximum bandwidth (in which case
      the link may be oversubscribed).
      This should be user-configurable; The default value should
      be the maximum bandwidth. the units are bytes per second.";
    leaf type {
      description "The maximum reservable bandwidth sub-tlv
        is tlv type 7,.";
      type uint8;
    }
    leaf length {
      description "The maximum reservable bandwidth sub-tlv is
        four octets in length.";
      type uint32;
    }
    leaf value {
      type uint32;
    }
  }
}
```

```
    }  
  }  
  
  container unreserved-bandwidth-stlv {  
    description "The unreserved bandwidth sub-tlv specifies  
      the amount of bandwidth not yet reserved at each of the  
      eight priority levels in IEEE floating point format.  
      The values correspond to the bandwidth that  
      can be reserved with a setup priority of 0 through 7,  
      arranged in increasing order with priority 0 occurring  
      at the start of the sub-tlv, and priority 7 at the end  
      of the sub-tlv. The initial values (before any bandwidth  
      is reserved) are all set to the maximum reservable  
      bandwidth. each value will be less than or  
      equal to the maximum reservable bandwidth.  
      The units are bytes per second.";  
    leaf type {  
      description "The unreserved bandwidth sub-tlv is  
        tlv type 8.";  
      type uint8;  
    }  
    leaf length {  
      description "The unreserved bandwidth sub-tlv is  
        32 octets in length.";  
      type uint32;  
    }  
    leaf value {  
      type uint32;  
    }  
  }  
  
  container administrative-group-stlv {  
    description "The administrative group sub-tlv contains  
      a 4-octet bit mask assigned by the network administrator.  
      Each set bit corresponds to one administrative group assigned  
      to the interface. a link may belong to multiple groups.  
      by convention, the least significant bit is referred to  
      as 'group 0', and the most significant bit is referred  
      to as 'group 31'. The administrative group is also  
      called resource class/color [5]..";  
  
    leaf type {  
      description "The administrative group sub-tlv is tlv type 9.";  
      type uint8;  
    }  
    leaf length {  
      description "The administrative group sub-tlv is  
        four octet in length.";
```



```

        type uint32;
    }
    leaf value {
        type uint32;
    }
}
}
}

grouping ospf-v3-router-lsa {
    container ospf-v3-router-lsa {
        description
            "router-lsas have ls type equal to 0x2001.
            Each router in an area originates one or more
            router-lsas. the complete collection of
            router-lsas originated by the router describe
            the state and cost of the router's interfaces
            to the area.";
        leaf option {
            description " 0 |nt|x|v|e|b| options .";
            type uint16;
            mandatory true;
        }
        list link-list{
            key "link-type interface-id neighbor-interface-id";
            leaf link-type {
                type enumeration {
                    enum "p2p";
                    enum "transit";
                    enum "reserved";
                    enum "virtual";
                }
                mandatory true;
            }
            leaf metric {
                description "The cost of using this router
                    interface for outbound traffic.";
                type uint32;
            }
            leaf interface-id {
                description "The interface id assigned to the
                    interface being described.";
                type uint32;
            }

            leaf neighbor-interface-id{
                description "The interface id the neighbor router
                    has associated with the link, as advertised in the

```

```

        neighbor's hello packets.  for transit (type
        2) links, the link's designated router is the
        neighbor described. For other link types, the
        sole adjacent neighbor is described.";
        type uint32;
    }
    leaf neighbor-router-id{
        description "The router id the of the neighbor router.
        For transit (type 2) links, the link's designated
        router is the neighbor described. For other link types,
        the sole adjacent neighbor is described.";
        type inet:ipv4-address;
    }
}
}
}

grouping ospf-v3-network-lsa {
    container ospf-v3-network-lsa {
        leaf option {
            description " 0 | options  .";
            type uint32;
            mandatory true;
        }
        list link-list{
            key "attached-router-id";
            leaf attached-router-id{
                description "The router ids of each of the routers
                attached to the link.  Actually, only those routers
                that are fully adjacent to the designated router
                are listed.  the designated router includes
                itself in this list.";
                type inet:ipv4-address;
            }
        }
    }
}

grouping ospf-v3-inter-area-prefix-lsa {
    container ospf-v3-inter-area-prefix-lsa {
        description " These lsas are the ipv6 equivalent of ospf
        for ipv4's type 3 summary-lsas (see section 12.4.3 of
        [ospfv2]).  originated by area border routers, they
        describe routes to ipv6 address prefixes that belong
        to other areas. A separate inter-area-prefix-lsa is originated
        for each ipv6 address prefix. ";
        leaf metric {
            description "The cost of this rout.";

```

```
        type uint32;
    }
    leaf prefix-length {
        type uint8;
        mandatory true;
    }
    leaf prefix-options {
        type uint8;
        mandatory true;
    }
    list address-prefix-list{
        key "address-prefix";
        leaf address-prefix{
            type inet:ipv6-prefix;
        }
    }
}

grouping ospf-v3-inter-area-router-lsa {
    container ospf-v3-inter-area-router-lsa {
        description " inter-area-router-lsas have ls
            type equal to 0x2004.  these lsas are the ipv6
            equivalent of ospf for ipv4's type 4 summary-lsas (see
            section 12.4.3 of [ospfv2]).  originated by
            area border routers, they describe routes
            to as boundary routers in other areas .";
        leaf options {
            type uint8;
            mandatory true;
        }
        leaf metric {
            description "The cost of  this rout.";
            type uint32;
        }
        leaf destination-router-id {
            description "The router id of the router being
                described by the lsa.";
            type inet:ipv4-address;
        }
    }
}

grouping ospf-v3-as-external-lsa {
    container ospf-v3-as-external-lsa {
        description " As-external-lsas have ls type equal to 0x4005.
            These lsas are originated by as boundary routers and describe
            destinations external to the as.  Each lsa describes a route
```

```
    to a single ipv6 address prefix. .";
  leaf options {
    type uint16;
    mandatory true;
  }
  leaf metric {
    description "The cost of this rout.";
    type uint16;
    mandatory true;
  }
  leaf prefix-length {
    type uint8;
    mandatory true;
  }
  leaf prefix-options {
    type uint8;
    mandatory true;
  }
  leaf referenced-ls-type {
    type uint8;
    mandatory true;
  }
  list address-prefix-list{
    key "address-prefix";
    leaf address-prefix{
      type inet:ipv6-prefix;
    }
  }
  leaf forwarding-address {
    type inet:ipv6-prefix;
    mandatory false;
  }
  leaf external-route-tag {
    type uint32;
    mandatory false;
  }
  leaf referenced-link-state-id {
    type uint32;
    mandatory false;
  }
}

grouping ospf-v3-nssa-lsa {
  container ospf-v3-nssa-lsa {
    description " Nssa-lsas have ls type equal to 0x4005.
      These lsas are originated by as boundary routers and
      describe destinations external to the as. Each lsa
```

```
    describes a route to a single ipv6 address prefix. .";
  leaf options {
    type uint16;
    mandatory true;
  }
  leaf metric {
    type uint16;
    mandatory true;
  }
  leaf prefixlength {
    type uint8;
    mandatory true;
  }
  leaf prefixoptions {
    type uint8;
    mandatory true;
  }
  leaf referenced-ls-type {
    type uint8;
    mandatory true;
  }
  list address-prefix-list {
    key "address-prefix";
    leaf address-prefix {
      type inet:ipv6-prefix;
    }
  }
  leaf forwarding-address {
    type inet:ipv6-prefix;
    mandatory false;
  }
  leaf external-route-tag {
    type uint32;
    mandatory false;
  }
  leaf referenced-link-state-id {
    type uint32;
    mandatory false;
  }
}

grouping ospf-v3-link-lsa {
  container ospf-v3-link-lsa {
    description " Link-lsas have ls type equal to 0x0008.
      A router originates a separate link-lsa for each
      attached physical link. These lsas have
      link-local flooding scope; they are never flooded
```

```
    beyond the associated link.";

    leaf priority {
        description " The router priority of the interface
            attaching the originating router to the link .";
        type uint8;
        mandatory true;
    }
    leaf options {
        description "The set of options bits that the router
            would like set in the network-lsa that will be
            originated by the designated router on
            broadcast or nbma links .";
        type uint32;
        mandatory true;
    }

    leaf link-local-interface-address {
        description "The originating router's link-local
            interface address on the link.";
        type inet:ipv6-address;
    }

    leaf prefixes {
        description "The number of ipv6 address prefixes contained
            in the lsa.";
        type uint32;
        mandatory true;
    }

    list address-prefix-list{
        key "address-prefix-index";
        leaf address-prefix-index{
            type uint32;
            mandatory true;
        }
        leaf prefix-length{
            type uint8;
            mandatory true;
        }
        leaf prefix-options{
            type uint8;
        }
        list address-prefix{
            key "address";
            leaf address{
                type inet:ipv6-prefix;
            }
        }
    }
}
```

```

    }
  }
}

grouping ospf-v3-intra-area-prefix-lsa {
  container ospf-v3-intra-area-prefix-lsa {
    description " Intra-area-prefix-lsas have ls
      type equal to 0x2009.  a router uses
      intra-area-prefix-lsas to advertise one
      or more ipv6 address prefixes that are associated
      with a local router address,
      an attached stub network segment, or an attached
      transit network segment.  In ipv4,
      the first two were accomplished via the router's
      router-lsa and the last via a network-lsa.
      In ospf for ipv6, all addressing information
      that was advertised in router-lsas and network-lsas
      has been removed and is now advertised in
      intra-area-prefix-lsas.";

    leaf prefixes {
      description "The number of ipv6 address prefixes
        contained in the lsa.";
      type uint32;
      mandatory true;
    }
    leaf referenced-ls-type {
      description " Referenced ls type, referenced link state id,
        and referenced advertising router identifies the router-lsa
        or network-lsa with which the ipv6
        address prefixes should be associated.  if referenced ls
        type is 0x2001, the prefixes are associated with a
        router-lsa, referenced link state id should be 0,
        and referenced advertising router
        should be the originating router's router id.
        If referenced ls type is 0x2002, the prefixes
        are associated with a network-lsa, referenced link
        state id should be the interface id of the link's
        designated router, and referenced advertising router
        should be the designated router's router id.";
      type uint16;
      mandatory true;
    }
    leaf referenced-link-state-id {
      type uint32;
      mandatory true;
    }
  }
}

```

```
    }
    leaf referenced-advertising-router {
        type inet:ipv4-address;
        mandatory true;
    }

    list address-prefix-list{
        key "address-prefix-index";
        leaf address-prefix-index{
            type uint32;
        }
        leaf prefix-length{
            type uint8;
            mandatory true;
        }
        leaf prefix-options{
            type uint8;
            mandatory true;
        }
        list address-prefix{
            key "address";
            leaf address{
                type inet:ipv6-prefix;
            }
        }
    }
}

grouping ospf-v3-te-router-ipv6-address {
    container ospf-v3-te-router-ipv6-address {
        description "The router ipv6 address tlv has
            type 3, length 16, and a value
            containing a 16-octet local ipv6 address.
            A link-local address must not be specified for this tlv.
            It must appear in exactly one traffic
            engineering lsa originated by an ospfv3 router supporting
            the te extensions.  the router ipv6 address tlv
            is a top-level tlv as defined in traffic engineering
            extensions to ospf ";
        leaf type {
            description "The router address tlv is type 3, has a
                length of 16.";
            type uint8;
            mandatory true;
        }
        leaf length {
            description "The router address tlv has a length of 4.";
            type uint16;
        }
    }
}
```



```

        mandatory true;
    }
    leaf router-id {
        description "The value of router address tlv is the
            16 octet ip address..";
        type inet:ipv6-address;
        mandatory true;
    }
}
}

```

```

container ospf-v4ur-instance {
    uses ospf-instance-commom;
    container mt-list {
        list multi-topo {
            key "mt-id";
            max-elements "unbounded";
            min-elements "1";
            uses ospf-mt-commom;
            container mt-rib {
                list route {
                    key "prefix";
                    max-elements "unbounded";
                    min-elements "0";
                    leaf prefix {
                        type inet:ipv4-prefix;
                        mandatory true;
                    }
                }
                container nexthop-list {
                    list nexthop {
                        key "ospf-nexthop";
                        max-elements "unbounded";
                        min-elements "0";
                        leaf ospf-nexthop {
                            type inet:ipv4-prefix;
                        }
                    }
                }
                leaf back-nexthop {
                    type inet:ipv4-prefix;
                }
            }
            uses ospf-route-commom;
        }
    }
}

```

```

container area-list {
    list area {

```

```
key "area-id";
max-elements "unbounded";
min-elements "1";
uses ospf-area-commom;
container lsdb {
  list lsa {
    key "lsa-v2-type link-state-id advertiser-id";
    max-elements "unbounded";
    min-elements "0";
    uses ospf-v2-lsa-header-commom;
    choice ls-type {
      case ospf-v2-router-lsa {
        uses ospf-v2-router-lsa;
      }

      case ospf-v2-network-lsa {
        uses ospf-v2-network-lsa ;
      }

      case ospf-v2-summary-lsa {
        uses ospf-v2-summary-lsa ;
      }

      case ospf-v2-as-external-lsa {
        uses ospf-v2-as-external-lsa ;
      }

      case ospf-v2-nssa-external-lsa {
        uses ospf-v2-nssa-external-lsa ;
      }

      case ospf-v2-te-router-lsa {
        uses ospf-v2-te-router-lsa ;
      }

      case ospf-te-link-lsa {
        uses ospf-te-link-lsa ;
      }
    }
  }
}

container interface-list {
  list interface {
    key "interface-index";
    max-elements "unbounded";
    min-elements "1";
    uses ospf-interface-commom;
```

```
    leaf ip-address {
      type inet:ipv4-address;
    }
    container nbr-list {
      list nbr {
        key "router-id";
        uses ospf-nbr-commom;
        leaf nbr-address {
          type inet:ipv4-address;
        }
        leaf ip-address {
          type inet:ipv4-address;
        }
      }
    }
  }
}

list network-list {
  description " configure the ospf .";
  key "network-prefix mask";
  leaf network-prefix {
    type inet:ipv4-prefix;
    mandatory true;
  }
  leaf mask {
    type inet:ipv4-prefix;
    mandatory true;
  }
}

list route-info-list {
  description " collision detection .";
  key "route-info-index";
  leaf route-info-index {
    type uint32;
    mandatory true;
  }
}

leaf router-id {
  type inet:ipv4-address;
  mandatory true;
}

list ip-address-list {
  description " collision detect .";
  key "ip-address";
  leaf ip-address {
    type inet:ipv4-address;
    mandatory true;
  }
}
```

}

```

container ospf-v6ur-instance {
    uses ospf-instance-commom;
    container mt-list {
        list multi-topo {
            key "mt-id";
            max-elements "unbounded";
            min-elements "1";
            uses ospf-mt-commom;
            container mt-rib {
                list route {
                    key "prefix";
                    max-elements "unbounded";
                    min-elements "0";
                    leaf prefix {
                        type inet:ipv6-prefix;
                        mandatory true;
                    }
                }
            }
            container nexthop-list {
                list nexthop {
                    key "ospf-nexthop";
                    max-elements "unbounded";
                    min-elements "0";
                    leaf ospf-nexthop {
                        type inet:ipv6-prefix;
                    }
                }
            }
            leaf back-nexthop {
                type inet:ipv6-prefix;
            }
            uses ospf-route-commom;
        }
    }
}

container area-list {
    list area {
        key "area-id";
        max-elements "unbounded";
    }
}

```

```
min-elements "1";
uses ospf-area-commom;
container lsdb {
  list lsa {
    key "lsa-v3-type link-state-id advertiser-id";
    max-elements "unbounded";
    min-elements "0";
    uses ospf-v3-lsa-header-commom;
    choice ls-type {
      case ospf-v3-router-lsa {
        uses ospf-v3-router-lsa ;
      }

      case ospf-v3-network-lsa {
        uses ospf-v3-network-lsa ;
      }
      case ospf-v3-inter-area-prefix-lsa {
        uses ospf-v3-inter-area-prefix-lsa ;
      }

      case ospf-v3-inter-area-router-lsa {
        uses ospf-v3-inter-area-router-lsa ;
      }

      case ospf-v3-as-external-lsa {
        uses ospf-v3-as-external-lsa ;
      }

      case ospf-v3-nssa-lsa {
        uses ospf-v3-nssa-lsa ;
      }

      case ospf-v3-link-lsa {
        uses ospf-v3-link-lsa ;
      }

      case ospf-v3-intra-area-prefix-lsa {
        uses ospf-v3-intra-area-prefix-lsa ;
      }

      case ospf-v3-te-router-ipv6-address-lsa {
        uses ospf-v3-te-router-ipv6-address ;
      }

      case te-link-lsa {
        uses ospf-te-link-lsa ;
      }
    }
  }
}
```

```

    }
  }

  container interface-list {
    list interface {
      key "interface-index";
      max-elements "unbounded";
      min-elements "1";
      uses ospf-interface-commom;
      leaf ip-address {
        type inet:ipv6-address;
        mandatory true;
      }
    }
    container nbr-list {
      list nbr {
        key "router-id";
        uses ospf-nbr-commom;
        leaf nbr-address {
          type inet:ipv6-address;
        }
        leaf ip-address {
          type inet:ipv6-address;
          mandatory true;
        }
      }
    }
  }
}

list network-list {
  description " Configure the ospf .";
  key "network-index";
  leaf network-index {
    type uint32;
    mandatory true;
  }
  leaf network-prefix {
    type inet:ipv4-prefix;
    mandatory true;
  }
  leaf mask {
    type inet:ipv4-prefix;
    mandatory true;
  }
}

list route-info-list {
  description " Collision detect .";
  key "route-info-index";

```

[illegible]

6. IANA Considerations

This draft registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC3688, the following registration is requested:

```
URI: urn:huawei:params:xml:ns:yang:rt:i2rs:ospf-protocol";
```

Registrant Contact: The I2RS WG of IETF

XML: N/A, the request URI is in the XML namespace.

This document registres a Yang module in the Yang Module Names registry [RFC6020] with the following information:

```
name: IETF-i2rs-ospf-protocol
```

```
namespace: urn:ietf:params:xml:ns:yang:rt:i2rs:ospf
```

```
prefix:ospf-protocol
```

reference: RFC XXXX

7. Security Considerations

This document introduces no new security threat over the security threats posed by security requirements as stated in [I-D.ietf-i2rs-architecture]. (The authors would like feedback on the security issues.)

8. Acknowledgements

TBD

9. References

9.1. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.
- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

9.2. Normative References

- [I-D.hares-i2rs-info-model-policy]
Hares, S. and W. Wu, "An Information Model for Basic Network Policy", draft-hares-i2rs-info-model-policy-03 (work in progress), July 2014.
- [I-D.hares-i2rs-usecase-reqs-summary]
Hares, S., "Summary of I2RS Use Case Requirements", draft-hares-i2rs-usecase-reqs-summary-00 (work in progress), July 2014.
- [I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-05 (work in progress), July 2014.

[I-D.ietf-i2rs-rib-info-model]

Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-03 (work in progress), May 2014.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Authors' Addresses

Lixing Wang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 10095
China

Email: wanglixing@huawei.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Nan Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: eric.wu@huawei.com

Internet
Internet-Draft
Intended status: Informational
Expires: April 17, 2015

D. Yeung
Y. Qu
Cisco Systems
J. Zhang
D. Bogdanovic
Juniper Networks
K. Sreenivasa
Brocade Communications System
October 14, 2014

Yang Data Model for OSPF Protocol
draft-yeung-netmod-ospf-02

Abstract

This document defines a YANG data model that can be used to configure and manage OSPF.

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 April 17, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Overview	2
1.1. Requirements Language	3
2. Design of Data Model	3
2.1. Overview	3
2.2. OSPFv2 and OSPFv3	5
2.3. Optional Features	5
2.4. Inheritance	5
2.5. OSPF Router Configuration	5
2.6. OSPF Instance Configuration	6
2.7. OSPF Area Configuration	8
2.8. OSPF Interface Configuration	9
2.9. OSPF notification	10
3. OSPF Yang Module	13
4. Security Considerations	69
5. Acknowledgements	69
6. References	69
6.1. Normative References	70
6.2. Informative References	71

1. Overview

YANG [RFC6020] is a data definition language that was introduced 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 of implementation for other interfaces, such as CLI and programmatic APIs.

A core routing data model is defined in [I-D.ietf-netmod-routing-cfg], and it proposes a basis for the development of data models for routing protocols. The interface data model is defined in [RFC7223] and is used for referencing interface from the routing protocol. This document defines a YANG data model that can be used to configure and manage OSPF and it is an augment to the core routing data model.

This document defines a YANG data model that can be used to configure and manage OSPF. Both OSPFv2 [RFC2328] and OSPFv3 [RFC5340] are supported. In addition to the core OSPF protocol, features described in different separate OSPF RFCs are also supported. They includes demand circuit [RFC1793], traffic engineering [RFC3630],

multiple address family [RFC5838], graceful restart [RFC3623] [RFC5187], NSSA [RFC3101] and sham link [RFC4577]. Those non-core features are made optional in the data model provided.

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. 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 different vendors. Differences are observed in term of how protocol engine is tied to routing domain, how multiple protocol engines could be instantiated and configuration inheritance, just to name a few.

The goal of this document is to define a data model that is capable of representing these differences. There is very little information that is designated as "mandatory", providing freedom to vendors to adapt this data model to their product implementation.

2.1. Overview

The OSPF YANG module defined in this document has all the common building blocks for OSPF protocol.

The OSPF YANG module augments the routing/routing-instance/routing-protocols/routing-protocol path of the ietf-routing module.

```

module: ospf
+--rw routing
  +--rw routing-instance [name]
    +--rw routing-protocols
      +--rw routing-protocol [name]
        +--rw ospf
          .
          .
          +--rw all-instances-inherit {instance-inheritance}?
          | .
          | .
          +--rw instance* [routing-instance af]
            .
            .
            +--rw all-areas-inherit {area-inheritance}?
            | .
            | .
            +--rw area* [area-id]
              .
              .
              +--rw all-interfaces-inherit {interface-inheritance}?
              | .
              | .
              +--rw interface* [interface]
                .
                .
            +--rw topology* [name]

```

The ospf is intended to match to the vendor specific OSPF configuration construct which is identified by a local identifier 'name'. The field 'version' allows support for OSPFv2 and OSPFv3.

The ospf container includes one or more OSPF protocol engines, each encapsulated in the instance entity. Each instance includes information for the routing domain it is running on based on the [routing-instance af] specification. There is no default routing domain assumed by the data model. For example, to enable OSPF on the default IPv4 routing domain of the vendor, this model requires an explicit instance entity with the specification like ["default" "ipv4-unicast"]. The instance also contains OSPF router level configuration

The instance/area and instance/area/interface container contain the OSPF configuration for the area and interface level respectively

The instance/topology container contain the OSPF configuration for topology when multi-topology feature is enabled

2.2. OSPFv2 and OSPFv3

The defined data model supports both OSPFv2 and OSPFv3.

The field 'version' is used to indicate the OSPF version and is a mandatory. Based on the version set, the data model change accordingly to accommodate the difference between the two versions.

2.3. Optional Features

Optional features are features beyond the basic of OSPF configurations and it is up to a vendor to decide the support of a particular feature on a particular device.

This module has declared a number of features, such as NSR, max-LSA etc.. It is intended that vendors will extend the features list.

2.4. Inheritance

This defined data model supports configuration inheritance for instances, areas and interfaces.

The all-instances-inherit, all-areas-inherit and all-interfaces-inherit containers provides a consistent way to configure inheritable command. Inheritance is treated as a feature. Vendors are expected to augment the above container to provide the list of inheritance command for their implementation.

2.5. OSPF Router Configuration

The container ospf is the top level container in this data model. It contains shared information among different OSPF instances under the container.

```
module: ospf
  +--rw ospf
    +--rw all-instances-inherit {instance-inheritance}?
      |   +--rw area
      |   +--rw interface
    +--rw operation-mode?          identityref
    +--rw instance* [routing-instance af]
      .
      .
```

2.6. OSPF Instance Configuration

The container instance represents an OSPF protocol engine. Each instance indicates the routing domain it is associated with based on [routing-instance af] and contains the router level configurations.

The all-areas-inherit container contains area configuration that could be inherited to all OSPF areas defined. Similarly, the all-areas-inherit also contains interface configuration that could be inherited to all the OSPF interfaces defined.

```

module: ospf

+--rw ospf
  .
  .
+--rw instance* [routing-instance af]
  +--rw routing-instance      rt:routing-instance-ref
  +--rw af                    identityref
  +--rw router-id?            yang:dotted-quad {router-id}?
  +--rw admin-distance
    | +--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 protocol-shutdown {protocol-shutdown}?
    | +--rw shutdown?        boolean
  +--rw auto-cost {auto-cost}?
    | +--rw enable?          boolean
    | +--rw reference-bandwidth? uint32
  +--rw maximum
    | +--rw paths?           uint16 {max-ecmp}?
    | +--rw max-lsa?         uint32 {max-lsa}?
  +--rw mpls
    | +--rw te-rid {te-rid}?
    | | +--rw (source)?
    | | | +---:(interface)
    | | | | +--rw interface?   if:interface-ref
    | | | +---:(explicit)
    | | | +--rw router-id?     inet:ipv4-address
    | +--rw ldp
    | | +--rw igp-sync?       boolean {ldp-igp-sync}?
    | | +--rw autoconfig?     boolean {ldp-igp-autoconfig}?
  +--rw all-areas-inherit {area-inheritance}?
    | +--rw area
    | +--rw interface

```


2.7. OSPF Area Configuration

The container area contains configurations of that area and the list of interface container represents all the OSPF interfaces active in the enclosing area.

```

module: ospf
  +--rw ospf
    .
    .
    +--rw instance* [routing-instance af]
      .
      .
      +--rw area* [area-id]
        +--rw area-id          area-id-type
        +--rw area-type?       identityref
        +--rw summary?         boolean
        +--rw default-cost?    uint32
        +--rw virtual-link* [router-id]
          +--rw router-id      yang:dotted-quad
          +--rw cost?          uint16
          +--rw hello-interval? uint16
          +--rw dead-interval?  uint16
          +--rw retransmit-interval? uint16
          +--rw transmit-delay? uint16
          +--rw mtu-ignore?     boolean {mtu-ignore}?
          +--rw lls?            boolean {lls}?
          +--rw prefix-suppression? boolean {prefix-suppression}?
          +--rw bfd?            boolean {bfd}?
          +--rw ttl-security {ttl-security}?
            +--rw enable?      boolean
            +--rw hops?        uint8
          +--rw protocol-shutdown {protocol-if-shutdown}?
            +--rw shutdown?    boolean
        +--rw sham-link* [local-id remote-id]
          +--rw local-id      inet:ip-address
          +--rw remote-id     inet:ip-address
          +--rw cost?         uint16
          +--rw hello-interval? uint16
          +--rw dead-interval?  uint16
          +--rw retransmit-interval? uint16
          +--rw transmit-delay? uint16
          +--rw mtu-ignore?     boolean {mtu-ignore}?
          +--rw lls?            boolean {lls}?
          +--rw prefix-suppression? boolean {prefix-suppression}?
          +--rw bfd?            boolean {bfd}?
          +--rw ttl-security {ttl-security}?
            +--rw enable?      boolean

```

```
| | | +--rw hops?      uint8
| | | +--rw protocol-shutdown {protocol-if-shutdown}?
| | | +--rw shutdown?  boolean
| | +--rw range* [prefix]
| | | +--rw prefix      inet:ip-prefix
| | | +--rw advertise?  boolean
| | | +--rw cost?       uint24
| | +--rw all-interfaces-inherit {interface-inheritance}?
| | +--rw interface
```

2.8. OSPF Interface Configuration

The container interface contains configurations of that interface.

The ospf-interfaces also contain interface configuration that could be inherited to all ospf-interface's defined.

```

module: ospf
+--rw ospf
  .
  .
+--rw instance* [routing-instance af]
  .
  .
+--rw area* [area-id]
  .
  .
+--rw interface* [interface]
  +--rw interface          if:interface-ref
  +--rw network-type?      enumeration
  +--rw passive?           boolean
  +--rw demand-circuit?    boolean {demand-circuit}?
  +--rw multi-area {multi-area-adj}?
  |   +--rw multi-area-id?  area-id-type
  |   +--rw cost?           uint16
  +--rw static-neighbors
  |   +--rw neighbor* [address]
  |   |   +--rw address      inet:ip-address
  |   |   +--rw cost?        uint16
  |   |   +--rw poll-interval? uint16
  |   |   +--rw priority?    uint8
  |   +--rw cost?           uint16
  +--rw hello-interval?    uint16
  +--rw dead-interval?     uint16
  +--rw retransmit-interval? uint16
  +--rw transmit-delay?    uint16
  +--rw mtu-ignore?        boolean {mtu-ignore}?
  +--rw lls?               boolean {lls}?
  +--rw prefix-suppression? boolean {prefix-suppression}?
  +--rw bfd?               boolean {bfd}?
  +--rw ttl-security {ttl-security}?
  |   +--rw enable?         boolean
  |   +--rw hops?           uint8
  +--rw protocol-shutdown {protocol-if-shutdown}?
  |   +--rw shutdown?       boolean
  +--rw topology* [name]
  |   +--rw name            rt:rib-ref
  |   +--rw cost?          uint32

```

2.9. OSPF notification

This YANG model defines a list of notifications to inform client of important events detected during the protocol operation. The notifications defined cover the common set of traps from OSPFv2 MIB [RFC4750] and OSPFv3 MIB [RFC5643].

```

module: ospf
notifications:
  +---n if-state-change
  |   +--ro routing-instance?      rt:routing-instance-ref
  |   +--ro routing-protocol-name? string
  |   +--ro instance-af
  |   |   +--ro af?  identityref
  |   +--ro link-type?            identityref
  |   +--ro interface
  |   |   +--ro interface?  if:interface-ref
  |   +--ro virtual-link
  |   |   +--ro area-id?      uint32
  |   |   +--ro neighbor-router-id? yang:dotted-quad
  |   +--ro sham-link
  |   |   +--ro area-id?      uint32
  |   |   +--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-instance?      rt:routing-instance-ref
  |   +--ro routing-protocol-name? string
  |   +--ro instance-af
  |   |   +--ro af?  identityref
  |   +--ro link-type?            identityref
  |   +--ro interface
  |   |   +--ro interface?      if:interface-ref
  |   |   +--ro packet-source?  yang:dotted-quad
  |   +--ro virtual-link
  |   |   +--ro area-id?      uint32
  |   |   +--ro neighbor-router-id? yang:dotted-quad
  |   +--ro sham-link
  |   |   +--ro area-id?      uint32
  |   |   +--ro local-ip-addr? inet:ip-address
  |   |   +--ro remote-ip-addr? inet:ip-address
  |   +--ro packet-type?          packet-type
  |   +--ro error?                enumeration
  +---n nbr-state-change
  |   +--ro routing-instance?      rt:routing-instance-ref
  |   +--ro routing-protocol-name? string
  |   +--ro instance-af
  |   |   +--ro af?  identityref
  |   +--ro link-type?            identityref
  |   +--ro interface
  |   |   +--ro interface?      if:interface-ref
  |   |   +--ro neighbor-router-id? yang:dotted-quad
  |   |   +--ro neighbor-ip-addr? yang:dotted-quad
  |   +--ro virtual-link
  |   |   +--ro area-id?      uint32

```

```

| |   +--ro neighbor-router-id?   yang:dotted-quad
+--ro sham-link
| |   +--ro area-id?               uint32
| |   +--ro local-ip-addr?         inet:ip-address
| |   +--ro neighbor-router-id?   yang:dotted-quad
| |   +--ro neighbor-ip-addr?     yang:dotted-quad
| +--ro state?                     nbr-state-type
+---n nbr-restart-helper-status-change
| +--ro routing-instance?         rt:routing-instance-ref
| +--ro routing-protocol-name?    string
| +--ro instance-af
| |   +--ro af? identityref
+--ro link-type?                  identityref
+--ro interface
| +--ro interface?               if:interface-ref
| +--ro neighbor-router-id?     yang:dotted-quad
| +--ro neighbor-ip-addr?       yang:dotted-quad
+--ro virtual-link
| +--ro area-id?                 uint32
| +--ro neighbor-router-id?     yang:dotted-quad
+--ro status?                     restart-helper-status-type
+--ro age?                         uint32
+--ro exit-reason?                restart-exit-reason-type
+---n rx-bad-packet
| +--ro routing-instance?         rt:routing-instance-ref
| +--ro routing-protocol-name?    string
| +--ro instance-af
| |   +--ro af? identityref
+--ro link-type?                  identityref
+--ro interface
| +--ro interface?               if:interface-ref
| +--ro packet-source?           yang:dotted-quad
+--ro virtual-link
| +--ro area-id?                 uint32
| +--ro neighbor-router-id?     yang:dotted-quad
+--ro sham-link
| +--ro area-id?                 uint32
| +--ro local-ip-addr?           inet:ip-address
| +--ro remote-ip-addr?          inet:ip-address
+--ro packet-type?                packet-type
+---n lsdb-approaching-overflow
| +--ro routing-instance?         rt:routing-instance-ref
| +--ro routing-protocol-name?    string
| +--ro instance-af
| |   +--ro af? identityref
+--ro ext-lsdb-limit?             uint32
+---n lsdb-overflow
| +--ro routing-instance?         rt:routing-instance-ref

```

```

|   +---ro routing-protocol-name?   string
|   +---ro instance-af
|   |   +---ro af?   identityref
|   +---ro ext-lsdb-limit?          uint32
+---n nssa-translator-status-change
|   +---ro routing-instance?        rt:routing-instance-ref
|   +---ro routing-protocol-name?   string
|   +---ro instance-af
|   |   +---ro af?   identityref
|   +---ro area-id?                 uint32
|   +---ro status?                  nssa-translator-state-type
+---n restart-status-change
|   +---ro routing-instance?        rt:routing-instance-ref
|   +---ro routing-protocol-name?   string
|   +---ro instance-af
|   |   +---ro af?   identityref
|   +---ro status?                  restart-status-type
|   +---ro restart-interval?        uint16
|   +---ro exit-reason?             restart-exit-reason-type

```

3. OSPF Yang Module

```

<CODE BEGINS>
module ospf {
  namespace "urn:ietf:params:xml:ns:yang:ospf";
  // replace with IANA namespace when assigned
  prefix ospf;

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-routing {
    prefix "rt";
  }

  organization
    "Cisco Systems
     170 West Tasman Drive
     San Jose, CA 95134-1706

```

```
    USA";
  contact
    "Derek Yeung myeung@cisco.com
    Yingzhen Qu yiqu@cisco.com
    Dean Bogdanovic deanb@juniper.net
    Jeffrey Zhang zzhang@juniper.net
    Kiran Agrahara Sreenivasa kkoushik@Brocade.com";

  description
    "This YANG module defines the generic configuration
    data for OSPF, which is common across all of the vendor
    implementations of the protocol. 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.

    Terms and Acronyms

    OSPF (ospf): Open Shortest Path First

    IP (ip): Internet Protocol

    IPv4 (ipv4): Internet Protocol Version 4

    IPv6 (ipv6): Internet Protocol Version 6

    MTU (mtu) Maximum Transmission Unit
    ";

  revision 2014-09-17 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for OSPF";
  }

  identity ospfv2 {
    base "rt:routing-protocol";
    description "OSPFv2";
  }

  identity ospfv3 {
    base "rt:routing-protocol";
    description "OSPFv3";
  }

  identity operation-mode {
```

```
    description
      "OSPF operation mode.";
  }

  identity ships-in-the-night {
    base operation-mode;
    description
      "Ships-in-the-night operation mode in which
      each OSPF instance carries only one address family";
  }

  identity area-type {
    description "Base identity for OSPF area type.";
  }

  identity normal {
    base area-type;
    description "OSPF normal area.";
  }

  identity stub {
    base area-type;
    description "OSPF stub area.";
  }

  typedef uint24 {
    type uint32 {
      range "0 .. 16777215";
    }
    description
      "24-bit unsigned integer.";
  }

  typedef area-id-type {
    type union {
      type uint32;
      type yang:dotted-quad;
    }
    description
      "Area ID 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 ExStart {
    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
          "Not attempted.";
      }
      enum InProgress {
        value "2";
        description
          "Restart in progress.";
      }
      enum Completed {
        value "3";
        description
          "Successfully completed.";
      }
      enum TimedOut {
        value "4";
        description
          "Timed out.";
      }
      enum TopologyChanged {
        value "5";
        description
          "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 acknowledgement 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.";
}

feature multi-topology {
    description
        "Support MTR.";
}

feature multi-area-adj {
    description
        "OSPF multi-area adjacency support as in RFC 5185.";
}

feature router-id {
    description
        "Set router ID per instance.";
}

feature demand-circuit {
    description
        "OSPF demand circuit support as in RFC 1793.";
}

feature mtu-ignore {
    description
        "Disable OSPF MTU mismatch detection on receiving
        DBD packets.";
}

feature lls {
    description
        "OSPF link-local signaling (LLS) as in RFC 5613.";
}

feature prefix-suppression {
```

```
    description
      "OSPF prefix suppression support as in RFC 6860.";
  }

  feature bfd {
    description
      "OSPF BFD support.";
  }

  feature ttl-security {
    description
      "OSPF ttl security check.";
  }

  feature nsr {
    description
      "Non-Stop-Routing (NSR).";
  }

  feature graceful-restart {
    description
      "Graceful OSPF Restart as defined in RFC3623 and RFC5187.";
  }

  feature protocol-shutdown {
    description
      "Shutdown the protocol.";
  }

  feature auto-cost {
    description
      "Calculate OSPF interface cost according to
       reference bandwidth.";
  }

  feature max-ecmp {
    description
      "Setting maximum number of ECMP paths.";
  }

  feature max-lsa {
    description
      "Setting maximum number of LSAs OSPF will receive.";
  }

  feature te-rid {
    description
      "TE router-id.;"
```

```
    }

    feature ldp-igp-sync {
      description
        "LDP IGP synchronization.";
    }

    feature ldp-igp-autoconfig {
      description
        "LDP IGP auto-config.";
    }

    feature protocol-if-shutdown {
      description
        "Shutdown the protocol over an interface.";
    }

    feature instance-inheritance {
      description
        "Support inheritance";
    }

    feature af-inheritance {
      description
        "Support inheritance";
    }

    feature area-inheritance {
      description
        "Support area inheritance";
    }

    feature interface-inheritance {
      description
        "Support interface inheritance";
    }

    grouping interface-common-config {
      description "Common configuration for all types of interfaces,
        including virtual link and sham link";

      leaf cost {
        type uint16 {
          range "1..65535";
        }
        description
          "Interface cost.";
      }
    }
  }
}
```

```
leaf hello-interval {
  type uint16 {
    range "1..65535";
  }
  units seconds;
  description
    "Time between hello packets.";
}

leaf dead-interval {
  type uint16 {
    range "1..65535";
  }
  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-interval'.";
  }
  description
    "Interval after which a neighbor is declared dead.";
}

leaf retransmit-interval {
  type uint16 {
    range "1..65535";
  }
  units seconds;
  description
    "Time between retransmitting unacknowledged Link State
    Advertisements (LSAs).";
}

leaf transmit-delay {
  type uint16 {
    range "1..65535";
  }
  units seconds;
  description
    "Estimated time needed to send link-state update.";
}

leaf mtu-ignore {
  if-feature mtu-ignore;
  type boolean;
  description
    "Enable/Disable ignoring of MTU in DBD packets.";
```



```
    }

    leaf lls {
      if-feature lls;
      type boolean;
      description
        "Enable/Disable link-local signaling (LLS) support.";
    }

    leaf prefix-suppression {
      if-feature prefix-suppression;
      type boolean;
      description
        "Suppress advertisement of the prefixes.";
    }

    leaf bfd {
      if-feature bfd;
      type boolean;
      description
        "Enable/disable bfd.";
    }

    container ttl-security {
      if-feature ttl-security;
      description "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 a OSPF packet may
           have traveled.";
      }
    }
  }
  container protocol-shutdown {
    if-feature protocol-if-shutdown;
    description
      "Protocol shutdown interface config state.";
    leaf shutdown {
      type boolean;
      description
        "Enable/Disable protocol shutdown on the interface.";
    }
  }
}
```

```
    }  
  }  
} // interface-common-config  
  
grouping interface-config {  
  description "Configuration for real interfaces.";   
  
  leaf network-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  
      "Network type.";  
  }  
  
  leaf passive {  
    type boolean;  
    description  
      "Enable/Disable passive.";  
  }  
  
  leaf demand-circuit {  
    if-feature demand-circuit;  
    type boolean;  
    description  
      "Enable/Disable demand circuit.";  
  }  
  
  container multi-area {  
    if-feature multi-area-adj;  
    description  
      "Configure ospf multi-area.";
```

```
    leaf multi-area-id {
      type area-id-type;
      description
        "Multi-area ID";
    }
    leaf cost {
      type uint16;
      description
        "Interface cost for multi-area.";
    }
  }
}

container static-neighbors {
  description "Static configured neighbors.";

  list neighbor {
    key "address";
    description
      "Specify a neighbor router.";

    leaf address {
      type inet:ip-address;
      description "Neighbor IP 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.";
    }

    leaf priority {
      type uint8 {
        range "1..255";
      }
      description "Neighbor priority for DR election.";
    }
  }
}

uses interface-common-config;
```

```
} // grouping interface-config

grouping tlv {
  description
    "TLV";
  leaf type {
    type uint16;
    description "TLV type.";
  }
  leaf length {
    type uint16;
    description "TLV length.";
  }
  leaf value {
    type yang:hex-string;
    description "TLV value.";
  }
}

grouping ospfv2-lsa-body {
  description "OSPFv2 LSA body.";
  container router {
    when "../../header/type = 1" {
      description
        "Only apply to Router-LSA.";
    }
    description
      "Router LSA.";
    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).";
        }
      }
    }
    description "Flags";
  }
  leaf num-of-links {
```

```
        type uint16;
        description "Number of links.";
    }
    list link {
        key "link-id link-data";
        description "Router LSA link.";
        leaf link-id {
            type union {
                type inet:ipv4-address;
                type yang:dotted-quad;
            }
            description "Link ID";
        }
        leaf link-data {
            type union {
                type inet:ipv4-address;
                type uint32;
            }
            description "Link data.";
        }
        leaf type {
            type uint8;
            description "Link type.";
        }
    }
    list topology {
        key "mt-id";
        description
            "Topology specific information.";
        leaf mt-id {
            type uint8;
            description
                "The MT-ID for topology enabled on the link.";
        }
        leaf metric {
            type uint16;
            description "Metric for the topology.";
        }
    }
}
}
container network {
    when "../header/type = 2" {
        description
            "Only apply to network LSA.";
    }
    description
        "Network LSA.";
    leaf network-mask {
```

```
    type inet:ipv4-address;
    description
      "The IP address mask for the network";
  }
  leaf-list attached-router {
    type yang:dotted-quad;
    description
      "List of the routers attached to the network.";
  }
}
container summary {
  when "../..//header/type = 3 or "
    + "../..//header/type = 4" {
    description
      "Only apply to Summary-LSA.";
  }
  description
    "Summary LSA.";
  leaf network-mask {
    type inet:ipv4-address;
    description
      "The IP address mask for the network";
  }
  list topology {
    key "mt-id";
    description
      "Topology specific information.";
    leaf mt-id {
      type uint8;
      description
        "The MT-ID for topology enabled on the link.";
    }
    leaf metric {
      type uint24;
      description "Metric for the topology.";
    }
  }
}
container external {
  when "../..//header/type = 5 or "
    + "../..//header/type = 7" {
    description
      "Only apply to AS-external-LSA and NSSA-LSA.";
  }
  description
    "External LSA.";
  leaf network-mask {
    type inet:ipv4-address;
```

```
        description
            "The IP address mask for the network";
    }
    list topology {
        key "mt-id";
        description
            "Topology specific information.";
        leaf mt-id {
            type uint8;
            description
                "The MT-ID for topology enabled on the link.";
        }
        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.";
        }
    }
}
container opaque {
    when "../header/type = 9 or "
        + "../header/type = 10 or "
        + "../header/type = 11" {
        description
            "Only apply to opaque LSA.";
    }
    description
        "Opaque LSA.";
```

```
list unknown-tlv {
  key "type";
  description "Unknown TLV.";
  uses tlv;
}

container router-address-tlv {
  leaf router-address {
    type inet:ipv4-address;
    description
      "Router address.";
  }
  description
    "Router address TLV.";
}

container 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.";
  }
  leaf-list local-if-ipv4-addr {
    type inet:ipv4-address;
    description
      "List of local interface IPv4 addresses.";
  }
  leaf-list local-remote-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 decimal64 {
      fraction-digits 2;
    }
  }
}
```



```
        description "Maximum bandwidth.";
    }
    leaf max-reservable-bandwidth {
        type decimal64 {
            fraction-digits 2;
        }
        description "Maximum reservable bandwidth.";
    }
    leaf unreserved-bandwidth {
        type decimal64 {
            fraction-digits 2;
        }
        description "Unreserved bandwidth.";
    }
    leaf admin-group {
        type uint32;
        description "Administrative group/Resource class/Color.";
    }
    list unknown-subtlv {
        key "type";
        description "Unknown sub-TLV.";
        uses tlv;
    }
    description
        "Link TLV.";
}
}

grouping ospfv3-lsa-options {
    description "OSPFv3 LSA options";
    leaf options {
        type bits {
            bit DC {
                description
                    "When set, the router support 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
            "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
                     readvertised by the translating NSSA area border.";
            }
            bit DN {
                description
                    "When set, the inter-area-prefix-LSA or
                     AS-external-LSA prefix has been advertised in a VPN
                     environment.";
            }
        }
    }
    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.";  
        }  
      }  
      description "Flags.";  
    }  
  
    leaf referenced-ls-type {  
      type uint16;  
      description "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 uint32;  
      description  
        "Referenced Link State ID.";  
    }  
  }  
  
  grouping ospfv3-lsa-body {
```

```
description "OSPFv3 LSA body.";
container router {
  when "../../../header/type = 8193" { // 0x2001
    description
      "Only apply to Router-LSA.";
  }
  description "Router LSA.";
  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.";
      }
    }
  }
  mandatory true;
  description "LSA option.";
}

uses ospfv3-lsa-options;

list link {
  key "interface-id neighbor-interface-id neighbor-router-id";
  description "Router LSA link.";
  leaf interface-id {
    type uint32;
    description "Interface ID.";
  }
  leaf neighbor-interface-id {
    type uint32;
    description "Neighbor Interface ID.";
  }
  leaf neighbor-router-id {
```

```
        type yang:dotted-quad;
        description "Neighbor Router ID";
    }
    leaf type {
        type uint8;
        description "Link type.";
    }
    leaf metric {
        type uint16;
        description "Metric.";
    }
}
}
container network {
    when "../..../header/type = 8194" { // 0x2002
        description
            "Only apply to network LSA.";
    }
    description "Network LSA.";

    uses ospfv3-lsa-options;

    leaf-list attached-router {
        type yang:dotted-quad;
        description
            "List of the routers attached to the network.";
    }
}
container inter-area-prefix {
    when "../..../header/type = 8195" { // 0x2003
        description
            "Only apply to inter-area-prefix LSA.";
    }
    leaf metric {
        type uint24;
        description "Metric";
    }
}

uses ospfv3-lsa-prefix;
description "Inter-Area-Prefix LSA.";
}
container inter-area-router {
    when "../..../header/type = 8196" { // 0x2004
        description
            "Only apply to inter-area-router LSA.";
    }
    uses ospfv3-lsa-options;
    leaf metric {
```

```
        type uint24;
        description "Metric";
    }
    leaf destination-router-id {
        type yang:dotted-quad;
        description
            "The Router ID of the router being described by the LSA.";
    }
    description "Inter-Area-Router LSA.";
}
container as-external {
    when "../..../header/type = 16389" { // 0x2007
        description
            "Only apply to as-external LSA.";
    }

    uses ospfv3-lsa-external;

    description "AS-External LSA.";
}
container nssa {
    when "../..../header/type = 8199" { // 0x2007
        description
            "Only apply to nssa LSA.";
    }

    uses ospfv3-lsa-external;

    description "NSSA LSA.";
}
container link {
    when "../..../header/type = 8" { // 0x0008
        description
            "Only apply to link LSA.";
    }
    leaf rtr-priority {
        type uint8;
        description "Router Priority of the interface.";
    }
}

uses ospfv3-lsa-options;

leaf link-local-interface-address {
    type inet:ipv6-address;
    description
        "The originating router's link-local
        interface address on the link.";
}
```

```
    leaf num-of-prefixes {
      type uint32;
      description "Number of prefixes.";
    }

    list prefix {
      key "prefix";
      description "List of prefixes associated with the link.";
      uses ospfv3-lsa-prefix;
    }
    description "Link LSA.";
  }
  container intra-area-prefix {
    when "../../header/type = 8201" { // 0x2009
      description
        "Only apply to intra-area-prefix LSA.";
    }
    description "Intra-Area-Prefix LSA.";

    leaf referenced-ls-type {
      type uint16;
      description "Referenced Link State type.";
    }
    leaf referenced-link-state-id {
      type uint32;
      description
        "Referenced Link State ID.";
    }
    leaf referenced-adv-router {
      type inet:ipv4-address;
      description
        "Referenced Advertising Router.";
    }
  }

  leaf num-of-prefixes {
    type uint16;
    description "Number of prefixes.";
  }
  list prefix {
    key "prefix";
    description "List of prefixes associated with the link.";
    uses ospfv3-lsa-prefix;
    leaf metric {
      type uint24;
      description "Metric";
    }
  }
}
```

```
    }

    grouping lsa-header {
      description
        "Common LSA for OSPFv2 and OSPFv3";
      leaf age {
        type uint16;
        mandatory true;
        description "LSA age.";
      }
      leaf type {
        type uint16;
        mandatory true;
        description "LSA type.";
      }
      leaf adv-router {
        type yang:dotted-quad;
        mandatory true;
        description "LSA advertising router.";
      }
      leaf seq-num {
        type uint32;
        mandatory true;
        description "LSA sequence number.";
      }
      leaf checksum {
        type uint16;
        mandatory true;
        description "LSA checksum.";
      }
      leaf length {
        type uint16;
        mandatory true;
        description "LSA length.";
      }
    }
  }

  grouping ospfv2-lsa {
    description
      "OSPFv2 LSA.";
    container header {
      description
        "Decoded OSPFv2 LSA header data.";
      leaf option {
        type bits {
          bit DC {
            description
              "When set, the router support demand circuits.";
          }
        }
      }
    }
  }
}
```



```
    }
    bit P {
      description
        "Only used in type-7 LSA. When set, the NSSA
        border router should translate the type-7 LSA
        to type-5 LSA.";
    }
    bit MC {
      description
        "When set, the router support MOSPF.";
    }
    bit E {
      description
        "This bit describes the way AS-external-LSAs
        are flooded";
    }
  }
  mandatory true;
  description "LSA option.";
}
leaf lsa-id {
  type inet:ipv4-address;
  mandatory true;
  description "LSA ID.";
}

leaf opaque-type {
  when "../header/type = 9 or "
    + "../header/type = 10 or "
    + "../header/type = 11" {
    description
      "Only apply to opaque LSA.";
  }
  type uint8;
  mandatory true;
  description "Opaque type.";
}

leaf opaque-id {
  when "../header/type = 9 or "
    + "../header/type = 10 or "
    + "../header/type = 11" {
    description
      "Only apply to opaque LSA.";
  }
  type uint24;
  mandatory true;
  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 "LSA ID.";
    }
    uses lsa-header;
  }
  container body {
    description
      "Decoded OSPF LSA body data.";
    uses ospfv3-lsa-body;
  }
}

grouping lsa {
  description
    "OSPF LSA.";
  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 as received/sent over the wire.";
  }
  choice version {
    description
      "OSPFv2 or OSPFv3 LSA body.";
    container ospfv2 {
```

```
        when "../../../rt:type = 'ospfv2'" {
            description "Applied to OSPFv2 only";
        }
        description "OSPFv2 LSA";
        uses ospfv2-lsa;
    }
    container ospfv3 {
        when "../../../rt:type = 'ospfv3'" {
            description "Applied to OSPFv3 only";
        }
        description "OSPFv3 LSA";
        uses ospfv3-lsa;
    }
}

grouping lsa-key {
    description
        "OSPF LSA key.";
    leaf lsa-id {
        type union {
            type inet:ipv4-address;
            type uint32;
        }
        description
            "LSA ID.";
    }
    leaf adv-router {
        type inet:ipv4-address;
        description
            "Advertising router.";
    }
}

grouping af-area-config {
    description
        "OSPF address-family specific area config state.";
    list range {
        key "prefix";
        description
            "Summarize routes matching address/mask (border
            routers 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
            "Cost of summary route.";
    }
}
}

grouping area-config {
    description
        "OSPF area config state.";
    leaf area-type {
        type identityref {
            base area-type;
        }
        default normal;
        description
            "Area type.";
    }

    leaf summary {
        when "area-type = 'stub' or area-type = 'nssa'" {
            description
                "Summary generation valid for stub/NSSA area.";
        }
        type boolean;
        description
            "Enable/Disable summary generation to the stub or
            NSSA area.";
    }

    leaf default-cost {
        when "area-type = 'stub' or area-type = 'nssa'" {
            description
                "Default cost for LSA advertised into stub or
                NSSA area.";
        }
        type uint32 {
            range "1..16777215";
        }
        description
            "Set the summary default-cost for a stub or NSSA area.";
    }
}
```

```
    }

    list virtual-link {
      key "router-id";
      description
        "OSPF virtual link";
      leaf router-id {
        type yang:dotted-quad;
        description
          "Virtual link router ID.";
      }

      uses interface-common-config;
    }

    list sham-link {
      key "local-id remote-id";
      description
        "OSPF sham link";
      leaf local-id {
        type inet:ip-address;
        description
          "Address of the local end-point.";
      }
      leaf remote-id {
        type inet:ip-address;
        description
          "Address of the remote end-point.";
      }
      uses interface-common-config;
    }

    uses af-area-config {
      when "../../../operation-mode = 'ospf:ships-in-the-night'" {
        description
          "Ships in the night configuration.";
      }
    }
  }
}

grouping instance-config {
  description
    "OSPF instance config state.";
  leaf router-id {
    if-feature router-id;
    type yang:dotted-quad;
    description
      "Defined in RFC 2328. A 32-bit number
```

```
        that uniquely identifies the router.";
    }

    container admin-distance {
        description "Admin distance config state.";
        choice granularity {
            description
                "Options for expressing admin distance
                 for intra-area and inter-area route";
            case detail {
                leaf intra-area {
                    type uint8;
                    description
                        "Admin distance for intra-area route.";
                }
                leaf inter-area {
                    type uint8;
                    description
                        "Admin distance for inter-area route.";
                }
            }
            case coarse {
                leaf internal {
                    type uint8;
                    description
                        "Admin distance for both intra-area and
                         inter-area route.";
                }
            }
        }
        leaf external {
            type uint8;
            description
                "Admin distance for both external route.";
        }
    }
}

container nsr {
    if-feature nsr;
    description
        "NSR config state.";
    leaf enable {
        type boolean;
        description
            "Enable/Disable NSR.";
    }
}
```

```
container graceful-restart {
  if-feature graceful-restart;
  description
    "Graceful restart config state.";
  leaf enable {
    type boolean;
    description
      "Enable/Disable graceful restart as defined in RFC 3623.";
  }
  leaf helper-enable {
    type boolean;
    description
      "Enable RestartHelperSupport in RFC 3623 Section B.2.";
  }
  leaf restart-interval {
    type uint16 {
      range "1..1800"; // Range is defined in RFC 3623.
    }
    units seconds;
    default "120"; // Default is defined in RFC 3623.
    description
      "RestartInterval option in RFC 3623 Section B.1.";
  }
  leaf helper-strict-lsa-checking {
    type boolean;
    description
      "RestartHelperStrictLSAChecking option in RFC 3623
      Section B.2.";
  }
}

container protocol-shutdown {
  if-feature protocol-shutdown;
  description
    "Protocol shutdown config state.";
  leaf shutdown {
    type boolean;
    description
      "Enable/Disable protocol shutdown.";
  }
}

container auto-cost {
  if-feature auto-cost;
  description
    "Auto cost config state.";
  leaf enable {
    type boolean;
```

```
        description
          "Enable/Disable auto cost.";
      }
      leaf reference-bandwidth {
        type uint32 {
          range "1..4294967";
        }
        units Mbits;
        description
          "Configure reference bandwidth in term of Mbits";
      }
    }
  }
}

container maximum {
  description
    "OSPF limits settings.";
  leaf paths {
    if-feature max-ecmp;
    type uint16 {
      range "1..32";
    }
    description
      "Maximum number of ECMP paths.";
  }
  leaf max-lsa {
    if-feature max-lsa;
    type uint32 {
      range "1..4294967294";
    }
    description
      "Maximum number of LSAs OSPF will receive.";
  }
}

container mpls {
  description
    "OSPF MPLS config state.";
  container te-rid {
    if-feature te-rid;
    description
      "Traffic Engineering stable IP address for system.";
    choice source {
      description
        "Different options for specifying TE router ID.";
      case interface {
        leaf interface {
          type if:interface-ref;
          description

```



```

        "Take the interface's IPv4 address as TE router ID.";
    }
}
case explicit {
    leaf router-id {
        type inet:ipv4-address;
        description
            "Explicitly configure the TE 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.";
    }
    leaf autoconfig {
        if-feature ldp-igp-autoconfig;
        type boolean;
        description
            "Enable LDP IGP interface auto-configuration.";
    }
}
}
}

grouping interface-operation {
    description
        "OSPF interface operation state.";
    reference "RFC2328 Section 9";
    uses interface-config;

    leaf state {
        type if-state-type;
        description "Interface state.";
    }

    leaf hello-timer {
        type uint32;
        units "milliseconds";
        description "Hello timer.";
    }
}

```

```
leaf wait-timer {
  type uint32;
  units "milliseconds";
  description "Wait timer.";
}

list neighbor {
  description
    "List of neighbors.";
  leaf neighbor-id {
    type leafref {
      path "../../neighbor/neighbor-id";
    }
    description "Neighbor.";
  }
}

leaf dr {
  type inet:ipv4-address;
  description "DR.";
}

leaf bdr {
  type inet:ipv4-address;
  description "BDR.";
}
} // interface-operation

grouping neighbor-operation {
  description
    "OSPF neighbor operation data.";

  leaf address {
    type inet:ip-address;
    description
      "Neighbor address.";
  }
  leaf dr {
    type inet:ipv4-address;
    description
      "Designated Router.";
  }
  leaf bdr {
    type inet:ipv4-address;
    description
      "Backup Designated Router.";
  }
  leaf state {
```

```
    type nbr-state-type;
    description
      "OSPF neighbor state.";
  }
}

grouping instance-operation {
  description
    "OSPF Address Family operation state.";
  leaf router-id {
    type yang:dotted-quad;
    description
      "Defined in RFC 2328. A 32-bit number
       that uniquely identifies the router.";
  }
}

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
  + "rt:routing-protocol" {
  when "rt:type = 'ospf:ospfv2' or rt:type = 'ospf:ospfv3'" {
    description
      "This augment is only valid for a routing protocol instance
       of OSPF (type 'ospfv2' or 'ospfv3').";
  }
  description "OSPF augmentation.";

  container ospf {
    description
      "OSPF.";

    container all-instances-inherit {
      if-feature instance-inheritance;
      description
        "Inheritance support to all instances.";
      container area {
        description
          "Area config to be inherited by all areas in
           all instances.";
      }
      container interface {
        description
          "Interface config to be inherited by all interfaces
           in all instances.";
      }
    }

    leaf operation-mode {
      type identityref {
```

```
        base operation-mode;
    }
    default ospf:ships-in-the-night;
    description
        "OSPF operation mode.";
}

list instance {
    key "routing-instance af";
    description
        "An OSPF routing protocol instance.";
    leaf routing-instance {
        type rt:routing-instance-ref;
        description
            "For protocol centric model, which is supported in
            default-instance only, this could reference any layer 3
            routing-instance.
            For routing-instance centric model, must reference the
            enclosing routing-instance.";
    }

    leaf af {
        type identityref {
            base rt:address-family;
        }
        description
            "Address-family of the instance.";
    }
}

uses instance-config;

container all-areas-inherit {
    if-feature area-inheritance;
    description
        "Inheritance for all areas.";
    container area {
        description
            "Area config to be inherited by all areas.";
    }
    container interface {
        description
            "Interface config to be inherited by all interfaces
            in 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;

    container all-interfaces-inherit {
      if-feature interface-inheritance;
      description
        "Inheritance for all interfaces";
      container interface {
        description
          "Interface config to be inherited by all
            interfaces.";
      }
    }

    list interface {
      key "interface";
      description
        "List of OSPF interfaces.";
      leaf interface {
        type if:interface-ref;
        description
          "Interface.";
      }
      uses interface-config;
    } // list of interfaces
  } // list of areas
} // list of instance
} // container ospf
}

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
  + "rt:routing-protocol/ospf:ospf/ospf:instance" {
  when "../rt:type = 'ospf:ospfv2' or ../rt:type = 'ospf:ospfv3'" {
    description
      "This augment is only valid for OSPF
        (type 'ospfv2' or 'ospfv3').";
  }
  if-feature multi-topology;
  description
    "OSPF multi-topology routing-protocol augmentation.";
}
```

```
list topology {
  // Topology must be in the same routing-instance
  // and of same AF as the container.
  key "name";
  description "OSPF topology.";
  leaf name {
    type rt:rib-ref;
    description "RIB";
  }
  list area {
    key "area-id";
    description
      "List of ospf areas";
    leaf area-id {
      type area-id-type;
      description
        "Area ID.";
    }
    uses area-config;
  }
}

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
  + "rt:routing-protocol/ospf:ospf/ospf:instance/"
  + "ospf:area/ospf:interface" {
  when "../..../rt:type = 'ospf:ospfv2'" {
    description
      "This augment is only valid for OSPFv2.";
  }
  if-feature ospf:multi-topology;
  description "OSPF multi-topology interface augmentation.";
  list topology {
    key "name";
    description "OSPF interface topology.";
    leaf name {
      type rt:rib-ref;
      description
        "One of the topology enabled on this interface";
    }
    leaf cost {
      type uint32;
      description
        "Interface cost for this topology";
    }
  }
}
```

```
augment "/rt:routing-state/rt:routing-instance/"
  + "rt:routing-protocols/rt:routing-protocol" {
    when "rt:type = 'ospf:ospfv2' or rt:type = 'ospf:ospfv3'" {
      description
        "This augment is only valid for a routing protocol instance
        of type 'ospfv2' or 'ospfv3'.";
    }
    description
      "OSPF configuration.";
    container ospf {
      description "OSPF";

      leaf operation-mode {
        type identityref {
          base operation-mode;
        }
        description
          "OSPF operation mode.";
      }

      list instance {
        key "routing-instance af";
        description
          "An OSPF routing protocol instance.";
        leaf routing-instance {
          type rt:routing-instance-ref;
          description
            "For protocol centric model, which is supported in
            default-instance only, this could reference any layer 3
            routing-instance.
            For routing-instance centric model, must reference the
            enclosing routing-instance.";
        }

        leaf af {
          type identityref {
            base rt:address-family;
          }
          description
            "Address-family of the instance.";
        }
      }

      uses instance-operation;

      list neighbor {
        key "area-id interface neighbor-id";
        description
          "List of OSPF neighbors.";
```

```
    leaf area-id {
      type area-id-type;
      description
        "Area ID.";
    }
    leaf interface {
      // Should it refer to config state leaf?
      type if:interface-ref;
      description
        "Interface.";
    }
    leaf neighbor-id {
      type inet:ipv4-address;
      description
        "Neighbor ID.";
    }
  }

  uses neighbor-operation;
} // list of OSPF neighbors

list interface {
  key "area-id interface";
  description
    "List of OSPF interfaces.";

  leaf area-id {
    type area-id-type;
    description "Area ID.";
  }
  leaf interface {
    // Should it refer to config state leaf?
    type if:interface-ref;
    description "Interface.";
  }

  uses interface-operation;
} // list of OSPF interfaces

list area {
  key "area-id";
  description "List of OSPF areas";
  leaf area-id {
    type area-id-type;
    description "Area ID.";
  }
} // list of OSPF areas

container databases {
```



```
description
  "OSPF databases.";
list link-scope-lsas {
  when "../.../rt:type = 'ospfv3'" {
    description
      "Link scope LSA only exists in OSPFv3.";
  }
  key "area-id interface lsa-type";
  description "List OSPF link scope LSA databases";
  leaf area-id {
    type uint32; // Should it refer to config state leaf?
    description "Area ID.";
  }
  leaf interface {
    // Should it refer to config state leaf?
    type if:interface-ref;
    description "Interface.";
  }
  leaf lsa-type {
    type uint8;
    description "OSPF link scope LSA type.";
  }
  list link-scope-lsa {
    key "lsa-id adv-router";
    description "List of OSPF link scope LSAs";
    uses lsa-key;
    uses lsa;
  }
} // list link-scope-lsas

list area-scope-lsas {
  key "area-id lsa-type";
  description "List OSPF area scope LSA databases";
  leaf lsa-type {
    type uint8;
    description "OSPF area scope LSA type.";
  }
  leaf area-id {
    type uint32; // Should it refer to config state leaf?
    description "Area ID.";
  }
  list area-scope-lsa {
    key "lsa-id adv-router";
    description "List of OSPF area scope LSAs";
    uses lsa-key;
    uses lsa;
  }
} // list area-scope-lsas
```

```
    list as-scope-lsas {
      key "lsa-type";
      description "List OSPF AS scope LSA databases";
      leaf lsa-type {
        type uint8;
        description "OSPF AS scope LSA type.";
      }
      list as-scope-lsa {
        key "lsa-id adv-router";
        description "List of OSPF AS scope LSAs";
        uses lsa-key;
        uses lsa;
      }
    } // list as-scope-lsas
  } // container databases
} // container ospf
}

augment "/rt:routing-state/rt:routing-instance/"
  + "rt:routing-protocols/rt:routing-protocol/"
  + "ospf:ospf/ospf:instance" {
  when "../rt:type = 'ospf:ospfv2'" {
    description
      "This augment is only valid for OSPFv2.";
  }
  if-feature multi-topology;
  description
    "OSPF multi-topology routing-protocol augmentation.";
  list topology {
    // Topology must be in the same routing-instance
    // and of same AF as the container.
    key "name";
    description "OSPF topology.";
    leaf name {
      type rt:rib-ref;
      description "RIB";
    }
  }
  list area {
    key "area-id";
    description
      "List of ospf areas";
    leaf area-id {
      type area-id-type;
      description
        "Area ID.";
    }
  }
}
```

```
    }  
  }  
  
  augment "/rt:routing-state/rt:routing-instance/"  
    + "rt:routing-protocols/rt:routing-protocol/"  
    + "ospf:ospf/ospf:instance/ospf:interface" {  
    when "../..//rt:type = 'ospf:ospfv2'" {  
      description  
        "This augment is only valid for OSPFv2.";  
    }  
    if-feature ospf:multi-topology;  
    description "OSPF multi-topology interface augmentation.";  
    list topology {  
      key "name";  
      description "OSPF interface topology.";  
      leaf name {  
        type rt:rib-ref;  
        description  
          "One of the topology enabled on this interface";  
      }  
    }  
  }  
}  
  
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 enumeration {  
      enum intra-area {  
        description "OSPF intra-area route";  
      }  
      enum inter-area {  
        description "OSPF inter-area route";  
      }  
      enum external-1 {  
        description "OSPF external route type 1";  
      }  
      enum external-2 {  
        description "OSPF External route type 2";  
      }  
    }  
  }  
}
```

```
    }
    enum nssa-1 {
      description "OSPF NSSA external route type 1";
    }
    enum nssa-2 {
      description "OSPF NSSA external route type 2";
    }
  }
  description "OSPF route type";
}
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
  when "rt:source-protocol = 'ospf:ospfv2' or "
    + "rt:source-protocol = 'ospf:ospfv3'" {
    description
      "This augment is only valid for a routes whose source
      protocol is OSPF.";
  }
  description
    "OSPF-specific route attributes.";
  uses route-content;
}

augment "/rt:active-route/rt:output/rt:route" {
  description
    "OSPF-specific route attributes in the output of 'active-route'
    RPC.";
  uses route-content;
}

identity if-link-type {
  description "Base identity for OSPF interface link type.";
}

identity if-link-type-normal {
  base if-link-type;
  description "OSPF interface link type normal.";
}

identity if-link-type-virtual-link {
  base if-link-type;
  description "OSPF interface link type virtual link.";
}

identity if-link-type-sham-link {
  base if-link-type;
  description "OSPF interface link type sham link.";
```

```
}

grouping notification-instance-hdr {
  description
    "This group describes common instance specific
    data for notifications.";

  leaf routing-instance {
    type rt:routing-instance-ref;
    description
      "Describe the routing instance.";
  }

  leaf routing-protocol-name {
    type string;
    description
      "Describes the name of the OSPF routing protocol.";
  }

  container instance-af {
    leaf af {
      type identityref {
        base rt:address-family;
      }
      description
        "Address-family of the instance.";
    }
    description
      "Describes the address family of the OSPF instance.";
  }
}

notification if-state-change {
  uses notification-instance-hdr;

  leaf link-type {
    type identityref {
      base if-link-type;
    }
    description "Type of OSPF interface.";
  }

  container interface {
    description "Normal interface.";
    leaf interface {
      type if:interface-ref;
      description "Interface.";
    }
  }
}
```

```
    }
    container virtual-link {
      description "virtual-link.";
      leaf area-id {
        type uint32;
        description "Area ID.";
      }
      leaf neighbor-router-id {
        type yang:dotted-quad;
        description "Neighbor router id.";
      }
    }
    container sham-link {
      description "sham-link.";
      leaf area-id {
        type uint32;
        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.";
      }
    }

    leaf state {
      type if-state-type;
      description "Interface state.";
    }

    description
      "This notification is sent when interface
      state change is detected.";
  }

  notification if-config-error {
    uses notification-instance-hdr;

    leaf link-type {
      type identityref {
        base if-link-type;
      }
      description "Type of OSPF interface.";
    }
  }
```

```
    container interface {
      description "Normal interface.";
      leaf interface {
        type if:interface-ref;
        description "Interface.";
      }
      leaf packet-source {
        type yang:dotted-quad;
        description "Source address.";
      }
    }
  }
  container virtual-link {
    description "virtual-link.";
    leaf area-id {
      type uint32;
      description "Area ID.";
    }
    leaf neighbor-router-id {
      type yang:dotted-quad;
      description "Neighbor router id.";
    }
  }
}

container sham-link {
  description "sham-link.";
  leaf area-id {
    type uint32;
    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.";
  }
}

leaf packet-type {
  type packet-type;
  description "OSPF packet type.";
}

leaf error {
  type enumeration {
    enum "badVersion" {
```

```
        description "Bad version";
    }
    enum "areaMismatch" {
        description "Area mismatch";
    }
    enum "unknownNbmaNbr" {
        description "Unknown NBMA neighbor";
    }
    enum "unknownVirtualNbr" {
        description "Unknown virtual link neighbor";
    }
    enum "authTypeMismatch" {
        description "Auth type mismatch";
    }
    enum "authFailure" {
        description "Auth failure";
    }
    enum "netMaskMismatch" {
        description "Network mask mismatch";
    }
    enum "helloIntervalMismatch" {
        description "Hello interval mismatch";
    }
    enum "deadIntervalMismatch" {
        description "Dead interval mismatch";
    }
    enum "optionMismatch" {
        description "Option mismatch";
    }
    enum "mtuMismatch" {
        description "MTU mismatch";
    }
    enum "duplicateRouterId" {
        description "Duplicate router ID";
    }
    enum "noError" {
        description "No error";
    }
}
description "Error code.";
}
description
    "This notification is sent when interface
    config error is detected.";
}

notification nbr-state-change {
    uses notification-instance-hdr;
```



```
leaf link-type {
  type identityref {
    base if-link-type;
  }
  description "Type of OSPF interface.";
}

container interface {
  description "Normal interface.";
  leaf interface {
    type if:interface-ref;
    description "Interface.";
  }
  leaf neighbor-router-id {
    type yang:dotted-quad;
    description "Neighbor router id.";
  }
  leaf neighbor-ip-addr {
    type yang:dotted-quad;
    description "Neighbor address.";
  }
}

container virtual-link {
  description "virtual-link.";
  leaf area-id {
    type uint32;
    description "Area ID.";
  }
  leaf neighbor-router-id {
    type yang:dotted-quad;
    description "Neighbor router id.";
  }
}

container sham-link {
  description "sham-link.";
  leaf area-id {
    type uint32;
    description "Area ID.";
  }
  leaf local-ip-addr {
    type inet:ip-address;
    description "Sham link local address.";
  }
  leaf neighbor-router-id {
    type yang:dotted-quad;
    description "Neighbor router id.";
  }
  leaf neighbor-ip-addr {
```

```
        type yang:dotted-quad;
        description "Neighbor address.";
    }
}

leaf state {
    type nbr-state-type;
    description "Neighbor state.";
}

description
    "This notification is sent when neighbor
    state change is detected.";
}

notification nbr-restart-helper-status-change {
    uses notification-instance-hdr;

    leaf link-type {
        type identityref {
            base if-link-type;
        }
        description "Type of OSPF interface.";
    }

    container interface {
        description "Normal interface.";
        leaf interface {
            type if:interface-ref;
            description "Interface.";
        }
        leaf neighbor-router-id {
            type yang:dotted-quad;
            description "Neighbor router id.";
        }
        leaf neighbor-ip-addr {
            type yang:dotted-quad;
            description "Neighbor address.";
        }
    }
}

container virtual-link {
    description "virtual-link.";
    leaf area-id {
        type uint32;
        description "Area ID.";
    }
    leaf neighbor-router-id {
        type yang:dotted-quad;
    }
}
```

```
        description "Neighbor router id.";
    }
}

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, if 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 neighbor restart
    helper status change is detected.";
}

notification rx-bad-packet {
    uses notification-instance-hdr;

    leaf link-type {
        type identityref {
            base if-link-type;
        }
        description "Type of OSPF interface.";
    }

    container interface {
        description "Normal interface.";
        leaf interface {
            type if:interface-ref;
            description "Interface.";
        }
        leaf packet-source {
            type yang:dotted-quad;
            description "Source address.";
        }
    }
}
```

```
    }
    container virtual-link {
      description "virtual-link.";
      leaf area-id {
        type uint32;
        description "Area ID.";
      }
      leaf neighbor-router-id {
        type yang:dotted-quad;
        description "Neighbor router id.";
      }
    }
  }

  container sham-link {
    description "sham-link.";
    leaf area-id {
      type uint32;
      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.";
    }
  }

  leaf packet-type {
    type packet-type;
    description "OSPF packet type.";
  }

  description
    "This notification is sent when an OSPF packet
    has been received on a interface that cannot be parsed.";
}

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 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
      ext-lsdb-limit.";
  }

  notification nssa-translator-status-change {
    uses notification-instance-hdr;

    leaf area-id {
      type uint32;
      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 ability to translate OSPF NSSA LSAs
      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 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 Acee Lindem, Yi Yang, Alexander Clemm, Gaurav Gupta, Ing-Wher Chen, Ladislav Lhotka and Stephane Litkowski for their thorough reviews and helpful comments.

This document was produced using Marshall Rose's xml2rfc tool.

6. References

6.1. Normative References

- [RFC1793] Moy, J., "Extending OSPF to Support Demand Circuits", RFC 1793, April 1995.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", RFC 3101, January 2003.
- [RFC3623] Moy, J., Pillay-Esnault, P., and A. Lindem, "Graceful OSPF Restart", RFC 3623, November 2003.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.
- [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, June 2006.
- [RFC4750] Joyal, D., Galecki, P., Giacalone, S., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, December 2006.
- [RFC5187] Pillay-Esnault, P. and A. Lindem, "OSPFv3 Graceful Restart", RFC 5187, June 2008.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.
- [RFC5643] Joyal, D. and V. Manral, "Management Information Base for OSPFv3", RFC 5643, August 2009.
- [RFC5838] Lindem, A., Mirtorabi, S., Roy, A., Barnes, M., and R. Aggarwal, "Support of Address Families in OSPFv3", RFC 5838, April 2010.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

6.2. Informative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management",
draft-ietf-netmod-routing-cfg-15 (work in progress), May
2014.

Authors' Addresses

Derek Yeung
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: myeung@cisco.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: yiqu@cisco.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: zzhang@juniper.net

Dean Bogdanovic
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

EMail: deanb@juniper.net

Kiran Agrahara Sreenivasa
Brocade Communications System
9442 Capital of Texas Hwy North
Arboretum Plaza One, Suite 500
Austin, TX 78759
USA

EMail: kkoushik@brocade.com