IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 21, 2018

H. Chen
S. Hares
Huawei Technologies
Y. Yang
Sockrate
Y. Fan
Casa Systems, Inc.
M. Toy
Verizon
Z. Li
China Mobile
L. Liu
Fujitsu

July 20, 2017

BGP for Communications among Controllers
draft-chen-idr-com-cntlrs-02

Abstract

This document describes extensions to the BGP routing protocol for supporting communications among SDN controllers in a centralized control system, which comprises multiple SDN controllers controlling a network with a number of domains.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 21, 2018.

Table of Contents

1. Introduction

A domain is a collection of network elements within a common sphere of address management or routing procedure which are operated by a single organization or administrative authority. Examples of such domains include IGP (OSPF or IS-IS) areas and Autonomous Systems.

A network running BGP is organized as multiple Autonomous Systems, each of which has a number of IGP areas.

The concepts of Software Defined Networks (SDN) have been shown to reduce the overall network CapEx and OpEx, whilst facilitating the deployment of services and enabling new features. The core principles of SDN include: centralized control to allow optimized usage of network resources and provisioning of network elements across domains.

For a network with a number of domains, it is natural to have multiple SDN controllers. each of the domains in the network is controlled by a controller. There are a few of architectures of controllers for achieving a centralized control on the network. One is a hierarchical architecture, another is a distributed architecture of controllers. A third one is a hybrid of a hierarchical and a distributed architecture of controllers.

At top level of a hierarchical architecture, it is a parent controller that is not a child controller. The parent controller controls a number of child controllers. Some of these child controllers are not parent controllers. Each of them controls a domain. Some other child controllers are also parent controllers, each of which controls multiple child controllers, and so on.

In a distributed architecture of controllers, there are a number of controllers, each of which controls a domain in a network with multiple domains and is adjacent to some of the other controllers. They control the network through their coordination.

This document describes extensions to the BGP routing protocol for supporting communications among SDN controllers in a centralized control system, which comprises multiple SDN controllers controlling a network with a number of domains.

2. Terminology

The following terminology is used in this document.

ABR: Area Border Router. Router used to connect two IGP areas (Areas in OSPF or levels in IS-IS).

ASBR: Autonomous System Border Router. Router used to connect together ASes of the same or different service providers via one or more inter-AS links.

BN: Boundary Node. A boundary node is either an ABR in the context of inter-area Traffic Engineering or an ASBR in the context of inter-AS Traffic Engineering. A Boundary Node is also called an Edge Node.

Entry BN of domain(n): a BN connecting domain(n-1) to domain(n) along the path found from the source node to the BN, where domain(n-1) is the previous hop (or upstream) domain of domain(n). An Entry BN is also called an in-BN or in-edge node.

Exit BN of domain(n): a BN connecting domain(n) to domain(n+1) along the path found from the source node to the BN, where domain(n+1) is the next hop (or downstream) domain of domain(n). An Exit BN is also called a out-BN or out-edge node.

Source Domain: For a tunnel from a source to a destination, the domain containing the source is the source domain for the tunnel.

Destination Domain: For a tunnel from a source to a destination, the domain containing the destination is the destination domain for the tunnel.

Source Controller: A controller controlling the source domain.

Destination Controller: A controller controlling the destination domain.

Parent Controller: A parent controller is a controller that communicates with a number of child controllers and controls a network with multiple domains through the child controllers. A BGP can be enhanced to be a parent controller.

Child Controller: A child controller is a controller that communicates with one parent controller and controls a domain in a network. A BGP can be enhanced to be a child controller.

Exception list: An exception list for a domain contains the nodes in the domain and its adjacent domains that are on the shortest path tree (SPT) that the parent controller is building.

GTID: Global Tunnel Identifier. It is used to identify a tunnel in a network.

GPID: Global Path Identifier. It is used to identify a path for a tunnel in a network.

Inter-area TE LSP: a TE LSP that crosses an IGP area boundary.

Inter-AS TE LSP: a TE LSP that crosses an AS boundary.

LSP: Label Switched Path

LSR: Label Switching Router

TED: Traffic Engineering Database.

This document uses terminology defined in [RFC2858].

3. Conventions Used in This Document

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

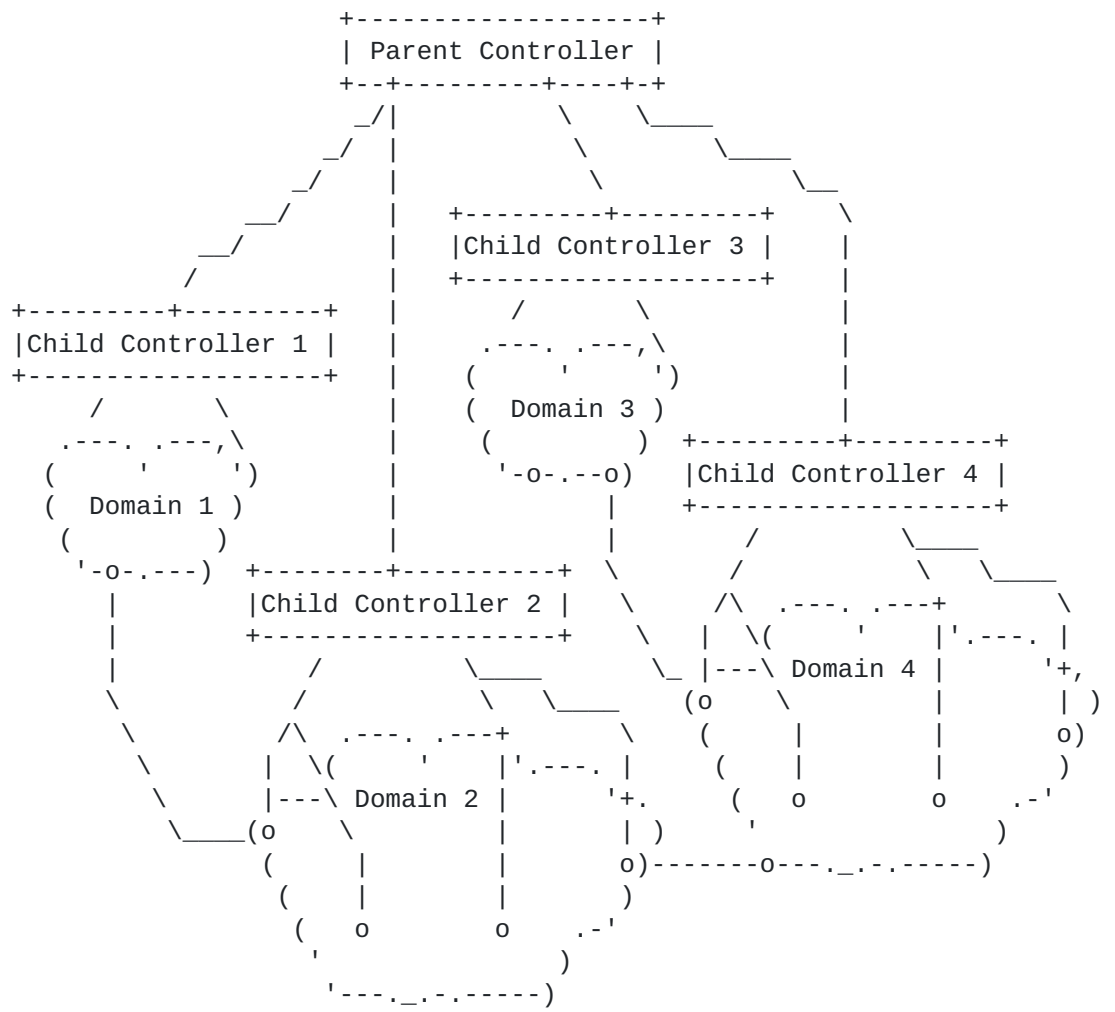
4. Requirements

This section summarizes the requirements for Hierarchical SDN Control System (need more text here).

5. Overview of SDN Control System

5.1. Hierarchical SDN Control System

The Figure below illustrates a hierarchical SDN control system. There is one Parent Controller and four Child Controllers: Child Controller 1, Child Controller 2, Child Controller 3 and Child Controller 4.



The parent controller communicates with its four child controllers and controls them, each of which controls (or is responsible for) a domain. Child controller 1 controls domain 1, Child controller 2 controls domain 2, Child controller 3 controls domain 3, and Child controller 4 controls domain 4.

One level of hierarchy of controllers is illustrated in the figure above. There is one parent controller at top level, which is not a child controller. Under the parent controller, there are four child controllers, which are not parent controllers.

In a general case, at top level there is one parent controller that is not a child controller, there are some controllers that are both parent controllers and child controllers, and there are a number of child controllers that are not parent controllers. This is a system

of multiple levels of hierarchies, in which one parent controller controls or communicates with a first number of child controllers, some of which are also parent controllers, each of which controls or communicates with a second number of child controllers, and so on.

The parent controller at top level has a level of 0. For a parent controller, which is a child controller of another parent controller having a level of N, this parent controller has a level of (N + 1).

Considering one parent controller and its child controllers, each of the child controllers controls a domain and has the topology information on the domain, the parent controller does not have the topology information on any domain controlled by a child controller normally. This is called a parent without domain topology.

In some special cases, the parent controller has the topology information on a region consisting of the domains controlled by its child controllers. In other words, the parent controller has the topology information on the domains controlled by its child controllers and the topology/inter-connections among these domains. This is called a parent with domain topology.

The parent controller receives requests for creating end to end tunnels from users or applications. For each request, the parent controller computes a path for the tunnel and creates the tunnel along the path.

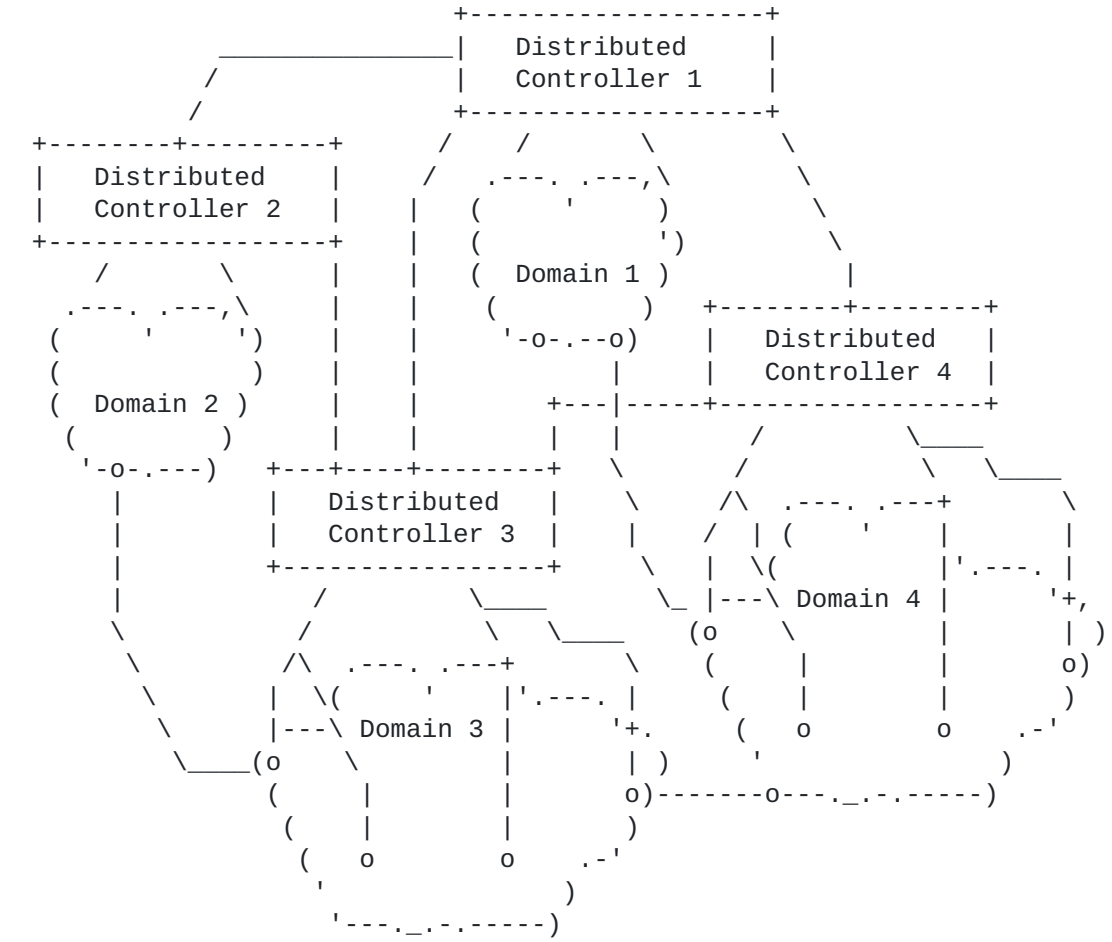
For a parent without domain topology, the parent controller asks each of its related child controllers to compute path segments from an entry boundary node to exit boundary nodes in the domain it controls or path segments from an exit boundary node in its domain to entry boundary nodes of other adjacent domains just using the inter-domain links attached to the exit boundary node. The parent controller builds a shortest path tree (SPT) using the path segments computed as links to get the end to end path and then creates the tunnel along the path by asking its related child controllers.

For a parent with domain topology, the parent controller computes a path for the tunnel using the topology information on the domains controlled by its child controllers. And then it creates the tunnel along the path computed through asking its related child controllers.

5.2. Distributed SDN Control System

The Figure below illustrates a distributed SDN control system. There are four Distributed Controllers: Distributed Controller 1, Distributed Controller 2, Distributed Controller 3 and Distributed Controller 4, which control Domain 1, Domain 2, Domain 3 and Domain 4

respectively.



The distributed controller 1 communicates with its adjacent distributed controllers, which are distributed controller 2, distributed controller 3 and distributed controller 4.

The distributed controller 2 communicates with its adjacent distributed controllers, which are distributed controller 1 and distributed controller 3.

The distributed controller 3 communicates with its adjacent distributed controllers, which are distributed controller 1, distributed controller 2 and distributed controller 4.

The distributed controller 4 communicates with its adjacent distributed controllers, which are distributed controller 1 and

distributed controller 3.

After receiving a request for creating an end to end tunnel from source A to destination Z for a given set of constraints, a distributed controller C1 computes an end to end path for the tunnel first. It sends another distributed controller C2 a message for building/growing a shortest path tree (SPT) (initially, SPT just has source A as root). This controller C2 computes a set of path segments in the domain it controls and continues to build/grow SPT using these path segments as links.

And then it creates the tunnel along the path computed through requesting the related distributed controllers.

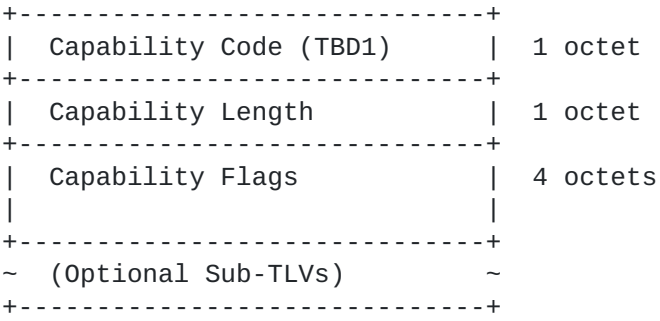
6. Extensions to BGP

This section describes extensions to BGP for supporting communications among SDN controllers in a SDN control system.

6.1. Capability and Controller Discovery

6.1.1. Capability Discovery

During/after a BGP session establishment between two BGP speakers, each of them advertises its capabilities for communications among SDN controllers (CSC) through the Open Message containing Capabilities Optional Parameter with a new triple (Capability Code, Capability Length, Capability Value) to indicate its capabilities for CSC. This new triple is called CSC capability triple. It has the following format.



The value of Capability Code is TBD1. The value of Capability Length is 4 octets plus the size of optional Sub-TLVs. The Capability Value comprises a Capability Flags field of 32 bits, which are numbered from the most significant as bit zero. Some of the capability flags are defined as follows.

- PS (Path Segments - 1 bit): Bit 0 is used as PS flag. It is set to 1 indicating support for computing path segments

- TS (Tunnel Segment - 1 bit): Bit 1 is used as TS flag. It is set to 1 indicating support for creating tunnel segment

- ET (End to end Tunnel - 1 bit): Bit 2 is used as ET flag. It is set to 1 indicating support for creation and maintenance of end to end LSP tunnels

- PC (Parent Controller - 1 bit): Bit 3 is used as PC flag. It is set to 1 indicating that the sender of the Open Message is a parent controller

- CC (Child Controller - 1 bit): Bit 4 is used as CC flag. It is set to 1 indicating that the sender of the Open Message is a child controller

- DC (Distributed Controller - 1 bit): Bit 5 is used as DC flag. It is set to 1 indicating that the sender of the Open Message is a distributed controller

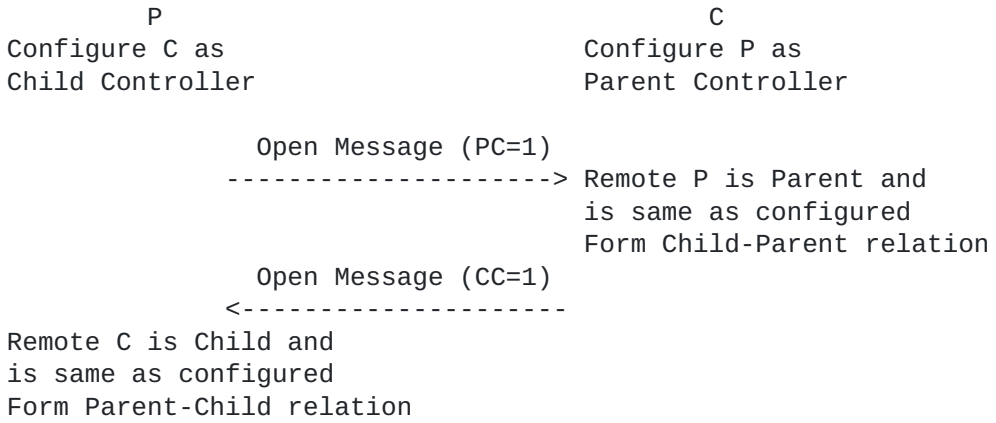
- Levels (Levels of Parent Controller - 4 bits): Bits 6 to 9 are used as Levels. It is set to the value of the levels of the local BGP speaker as a parent controller.

6.1.2. Controller Relation Discovery

6.1.2.1. Parent Child Relation Discovery

For a parent controller P and a child controller C connected by a BGP session and having a normal BGP peer adjacency, their parent-child relation is discovered through Open Messages exchanged between the parent controller and the child controller. The following is a sequence of events related to a controller relation discovery.

Controller P sends controller C an Open Message containing a CSC capability triple with parent flag PC set to 1 after controller C is configured as a child controller over a BGP session between P and C.



When C receives the Open Message from P and determines that PC=1 in the message is consistent with the parent controller configured locally, it forms Child-Parent relation between C and P. It sends controller P an Open Message containing a CSC capability triple with child controller flag CC set to 1 after controller P is configured as a parent controller over a BGP session between C and P.

When P receives the Open Message from C and determines that CC=1 in the message is consistent with the Child controller configured locally, it forms Parent-Child relation between P and C.

After the Parent-Child relation between P and C is formed, this relation is broken if the configuration "C as Child Controller" on parent controller P is deleted or "P as Parent Controller" on child controller C is removed.

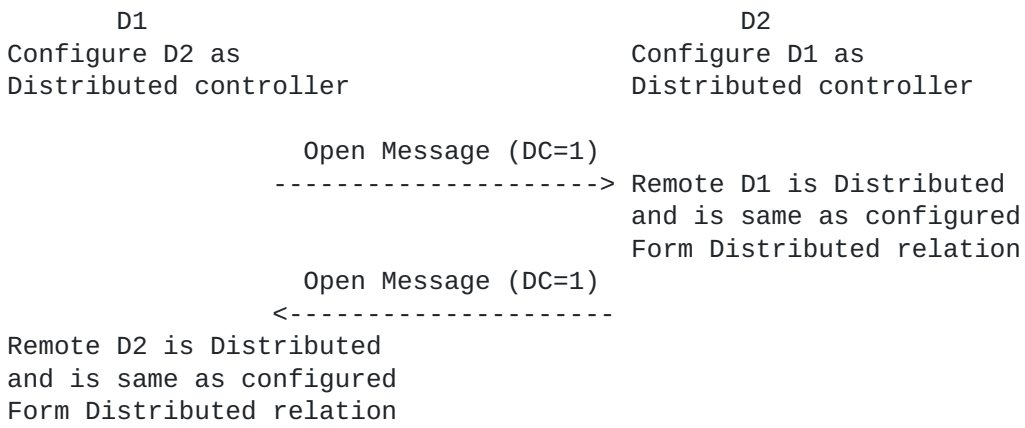
When the configuration "C as Child Controller" is deleted from parent controller P, P breaks/removes the Parent-Child relation between P and C and sends C an Open Message with PC = 0. When child controller C receives the Open Message with PC = 0 from P, it determines that the remote end P is no longer its parent controller as configured locally and breaks/removes the Child-Parent relation between C and P.

When the configuration "P as Parent Controller" is deleted from child controller C, C breaks/removes the Child-Parent relation between C and P and sends P an Open Message with CC = 0. When parent controller P receives the Open Message with CC = 0 from C, it determines that the remote end C is no longer its child controller as configured locally and breaks/removes the Parent-Child relation between P and C.

6.1.2.2. Distributed Relation Discovery

For a distributed controller D1 and another distributed controller D2 connected by a BGP session and having a normal BGP peer adjacency, their distributed controllers relation is discovered through Open Messages exchanged between D1 and D2. The following is a sequence of events related to a controller relation discovery.

Controller D1 sends controller D2 an Open Message containing a CSC capability triple with Distributed Controller flag DC set to 1 after controller D2 is configured as a distributed controller over a BGP session between D1 and D2.



When D2 receives the Open Message from D1 and determines that DC=1 in the message is consistent with the distributed controller configured locally, it forms distributed relation between D2 and D1. It sends controller D1 an Open Message containing a CSC capability triple with Distributed Controller flag DC set to 1 after controller D1 is configured as a distributed controller over a BGP session between D2 and D1.

When D1 receives the Open Message from D2 and determines that DC=1 in the message is consistent with the distributed controller configured locally, it forms distributed relation between D1 and D2.

When the configuration "D2 as Distributed Controller" is deleted from controller D1, D1 breaks/removes the distributed relation between D1 and D2 and sends D2 an Open Message with DC = 0. When controller D2 receives the Open Message with DC = 0 from D1, it determines that the remote end D1 is no longer its adjacent distributed controller as configured locally and breaks/removes the distributed relation between D2 and D1.

When the configuration "D1 as Distributed Controller" is deleted from controller D2, D2 breaks/removes the distributed relation between D2 and D1 and sends D1 an Open Message with DC = 0. When controller D1 receives the Open Message with DC = 0 from D2, it determines that the remote end D2 is no longer its adjacent distributed controller as configured locally and breaks/removes the distributed relation between D1 and D2.

6.2. Messages for SDN Control System

This section gives an overview of messages for a Hierarchical SDN Control System (HSCS) and a Distributed SDN Controller System (DSCS), which is followed by the description on the contents and semantics of the messages. Most of the messages for HSCS are similar to those for DSCS.

6.2.1. Overview of Messages

6.2.1.1. Overview of Messages for HSCS

Various types of messages for supporting HSCS are listed below.

Message for Connections and Accesses Advertisement: It is a message that a child controller sends its parent controller to describe the connections from the domain it controls to its adjacent domains and the access points in the domain to be accessible outside of the domain.

Request for Computing Path Segments: It is a message that a parent controller sends a child controller to request the child controller for computing path segments in the domain the child controller controls.

Reply for Computing Path Segments: It is a message that a child controller sends a parent controller to reply the parent controller for a request message for computing path segments after receiving the request message from the parent controller and computing path segments as requested, which normally contains the path segments computed.

Request for Removing Path Segments: It is a message that a parent controller sends a child controller to request the child controller for removing the path segments computed by the child controller and stored in the child controller.

Reply for Removing Path Segments: It is a message that a child controller sends a parent controller to reply the parent controller for a request message for removing a set of path segments after receiving the request message from the parent controller and removing the path segments as requested, which normally contains a status of removing path segments.

Request for Keeping Path Segments: It is a message that a parent controller sends a child controller to request the child controller for keeping a set of path segments computed by the child controller and stored in the child controller.

Reply for Keeping Path Segments: It is a message that a child controller sends a parent controller to reply the parent controller for a request message for keeping path segments after receiving the request message from the parent controller and keeping the path segments as requested, which normally contains a status of keeping path segments.

Request for Creating Tunnel Segment: It is a message that a parent controller sends a child controller to request the child controller for creating tunnel segments in the domain the child controller controls.

Reply for Creating Tunnel Segment: It is a message that a child controller sends a parent controller to reply the parent controller for a request message for creating tunnel segment after receiving the request message from the parent controller and creating tunnel segment as requested, which normally contains a status of creating tunnel segment and a label and an interface.

Request for Removing Tunnel Segment: It is a message that a parent controller sends a child controller to request the child controller for removing the tunnel segment created by the child controller.

Reply for Removing Tunnel Segment: It is a message that a child controller sends a parent controller to reply the parent controller for a request message for removing tunnel segment after receiving the request message from the parent controller and removing the tunnel segment as requested, which normally contains a status of removing tunnel segment.

Note that four messages Request and Reply for Removing Path Segments and Request and Reply for Keeping Path Segments are not needed if path segments computed are not stored/remembered by a controller. But in this case, the path segment in each domain along the end to end path computed needs to be re-computed when a tunnel along the

path is set up.

6.2.1.2. Overview of Messages for DSCS

Various types of messages for supporting DSCS are listed below.

Message for Connections and Accesses Advertisement: It is a message that a distributed controller sends its adjacent distributed controllers to describe the connections from the domain it controls to its adjacent domains and the access points in the domain to be accessible outside of the domain.

Request for Growing SPT: It is a message that a distributed controller D1 sends another distributed controller D2 to request D2 for growing a shortest path tree (SPT) (initially, SPT is empty). D2 computes a set of path segments in the domain it controls and builds/grows SPT using these path segments as links.

Reply for Growing SPT: It is a message that a distributed controller D1 sends another distributed controller D2 to reply D2 for a request message for Growing SPT after receiving the request message and growing SPT as requested.

Request for Removing Path Segments: It is a message that a distributed controller D1 sends another distributed controller D2 to request D2 for removing the path segments computed by D2 and stored in D2.

Reply for Removing Path Segments: It is a message that a distributed controller D1 sends another distributed controller D2 to reply D2 for a request message for removing a set of path segments after receiving the request message from D2 and removing the path segments as requested.

Request for Keeping Path Segments: It is a message that a distributed controller D1 sends another distributed controller D2 to request D2 for keeping a set of path segments computed by D2 and stored in D2.

Reply for Keeping Path Segments: It is a message that a distributed controller D1 sends another distributed controller D2 to reply D2 for a request message for keeping path segments after receiving the request message from D2 and keeping the path segments as requested.

Request for Creating Tunnel Segment: It is a message that a distributed controller D1 sends another distributed controller D2 to request D2 for creating tunnel segments in the domain D2 controls.

Reply for Creating Tunnel Segment: It is a message that a distributed controller D1 sends another distributed controller D2 to reply D2 for a request message for creating tunnel segment after receiving the request message and creating tunnel segment as requested, which normally contains a status of creating tunnel segment and a label and an interface.

Request for Removing Tunnel Segment: It is a message that a distributed controller D1 sends another distributed controller D2 to request D2 for removing the tunnel segment created by D2.

Reply for Removing Tunnel Segment: It is a message that a distributed controller D1 sends another distributed controller D2 to reply D2 for a request message for removing tunnel segment after receiving the request message from D2 and removing the tunnel segment as requested, which normally contains a status of removing tunnel segment.

6.2.2. Messages for Hierarchical SDN Control System

6.2.2.1. Message for Connections and Accesses Advertisement in HSCS

After a child controller discovers its parent controller, it sends its parent controller a message for connections and accesses advertisement.

A message for connections and accesses advertisement (CAAdv message for short) from a child controller comprises:

- Inter-domain links from the domain the child controller controls to its adjacent domains.
- The addresses in the domain to be accessible to the outside of the domain.
- Attributes of each of the boundary nodes of the domain.

6.2.2.2. Request for Computing Path Segments in HSCS

After receiving a request for creating an end to end tunnel from source A to destination Z for a given set of constraints, a parent controller allocates a global tunnel identifier (GTID) for the end to end tunnel crossing domains and a path identifier (PID) for an end to

end path to be computed for the tunnel. The parent controller sends a request message to each of its related child controllers for computing a set of path segments in the domain the child controller controls in a special order. The parent controller builds a shortest path tree (SPT) using these path segments and obtains a shortest path from source A to destination Z that satisfies the constraints.

A request message for computing path segments (CPSReq message for short) from a parent controller to a child controller comprises:

- The address or identifier of the start-node (saying X) in the domain controlled by the child controller. From this node, a number of path segments are to be computed.

- The tunnel ID information (Tunnel-ID-Info for short) containing the global tunnel identifier (GTID) and the path identifier (PID). For the path of the tunnel, which is identified by the Tunnel-ID-Info, a number of path segments are to be computed.

- An exception-list containing the nodes that are on the SPT and in the domain controlled by the child controller or its adjacent domains.

- The constraints for the path such as bandwidth constraints and color constraints.

- A destination node Z. If Z is in the domain controlled by the child controller, the child controller computes a shortest path segment satisfying the constraints from node X to node Z within the domain.

- Options for computing path segments:

  E: E set to 1 indicating computing a shortest path segment satisfying the constraints from node X to each of the edge nodes of the domain controlled by the child controller except for the nodes in the exception-list. E is set to 1 if there is not any previous hop of node X in the domain.

After receiving the request message, the child controller computes a shortest path segment satisfying the constraints from node X to each of the edge nodes of the domain controlled by the child controller except for the nodes in the exception-list if E is 1. In addition, it computes a shortest one hop path segment satisfying the constraints from node X to each of the edge nodes of the adjacent domains except for the edge nodes in the exception-list just using the inter-domain links attached to node X if node X is an edge node of the domain and an end point of an inter-domain link.

6.2.2.3. Reply for Computing Path Segments in HSCS

After receiving a request message from a parent controller for computing path segments, a child controller computes the path segments as requested in the message and sends the parent controller a reply message to reply the request message, which contains the path segments computed.

A reply message for computing path segments (CPSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID). For the path of the tunnel, which is identified by the Tunnel-ID-Info, a number of path segments are to be computed.

- The address or identifier of the start-node (saying X) in the domain controlled by the child controller. From this node, the path segments are computed.

- For each shortest path segment from node X to node Y computed, the address or identifier of node Y and the cost of the shortest path segment from node X to node Y.

The child controller stores the details about every shortest path segment computed under GTID and PID when it sends the reply message containing the path segments to the parent controller.

The child controller may delete the path segments computed for the GTID and PID if it does not receive any request for keeping them from the parent controller for a given period of time.

6.2.2.4. Request for Removing Path Segments in HSCS

After a shortest path satisfying a set of constraints from source A to destination Z is computed successfully, a parent controller may delete the path segments computed and stored in the related child controllers, which are not any part of the shortest path. When the path computation fails, the parent controller may delete the path segments computed and stored in the related child controllers. A parent controller may send a child controller a request message for removing the path segments computed by the child controller and stored in the child controller.

1). A request message for removing path segments (RPSReq message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID).

All the path segments stored under GTID in the child controller are to be removed.

2). A request message for removing path segments comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).

All the path segments stored under GTID and PID in the child controller are to be removed.

3). A request message for removing path segments comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID)

- A list of start-point (or node) addresses or identifiers.

All the path segments stored in the child controller under GTID and PID and with a start-point or node from the list of start-point (or node) addresses or identifiers are to be removed.

4). A request message for removing path segments comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID)

- A list of start-point (or node) addresses or identifiers

- A list of pairs (start-point, a list of end-points), which identifies the path segments from start-point of each pair to each of the end-points in the list of the pairs.

In addition to the path segments as described in the previous message, the path segments stored in the child controller under GTID and PID and identified by the list of pairs are to be removed.

6.2.2.5. Reply for Removing Path Segments in HSCS

After removing the path segments as requested by a request message for removing path segments from a parent controller, a child controller sends the parent controller a reply message for removing path segments.

A reply message for removing path segments (RPSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID)

- Status of the path segments removal:

  Success: The path segments requested for removal are removed successfully.

  Fail: The path segments requested for removal can not be removed.

- Error code and reasons for failure if the status is Fail.

6.2.2.6. Request for Keeping Path Segments in HSCS

After a shortest path satisfying a set of constraints from source A to destination Z is computed, a parent controller may send a request message for keeping path segments to each of the related child controllers to keep the path segments on the shortest path.

A request message for keeping path segments (KPSReq message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).

- A list of pairs (start-point, end-point), each of which identifies the path segment from the start-point of the pair to the end-point of the pair.

The child controller will keep the path segments given by the list of pairs (start-point, end-point) stored under GTID and PID. It will remove all the other path segments stored under GTID and PID.

6.2.2.7. Reply for Keeping Path Segments in HSCS

After keeping path segments as requested by a request message for keeping path segments from a parent controller, a child controller sends the parent controller a reply message for keeping path segments.

A reply message for keeping path segments (KPSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).

- Status of the path segment retention:

  Success: The path segments requested for retention are retained successfully.

  Fail: The path segments requested for retention can not be retained.

- Error code and reasons for failure if the status is Fail.

6.2.2.8. Request for Creating Tunnel Segment in HSCS

After obtaining the end to end shortest point to point (P2P) path, a parent controller creates a tunnel along the path crossing multiple domains through sending a request message for creating tunnel segment to each of the child controllers along the path in a reverse direction to create a tunnel segment.

A request message for creating tunnel segment (CTSReq message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).

- A path segment from a start-point to an end-point for parent without domain topology or a path segment details/ERO for parent with domain topology.

- A label and an interface if the domain controlled by the child control is not a destination domain.

For a parent without domain topology, the child controller allocates and reserves link bandwidth along the path segment identified by the start-point and end-point, assigns labels along the path segment, and writes cross connects on each of the nodes along the path segment.

For a parent with domain topology, the child controller assigns labels along the path segment ERO and writes cross connects on each of the nodes along the path segment. The link bandwidth along the path segment is allocated and reserved by the parent controller.

For the non destination domain, the child controller writes the cross connect on the edge node to the downstream domain using the label and the interface from the downstream domain in the message.

For the non source domain, the child controller will include a label and an interface in a message to be sent to the parent controller. The interface connects the edge node of the upstream domain along the

path. The label is allocated for the interface on the node that is the next hop of the edge node.

6.2.2.9. Reply for Creating Tunnel Segment in HSCS

After creating tunnel segment as requested by a request message for creating tunnel segment from a parent controller, a child controller sends the parent controller a reply message for creating tunnel segment.

A reply message for creating tunnel segment (CTSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).
- Status of the tunnel segment creation:

  Success: The tunnel segment requested is created successfully.

  Fail: The tunnel segments requested can not be created.
- A label and an interface if the domain controlled by the child controller is not source domain and the status is Success.
- Error code and reasons for failure if the status is Fail.

For the non source domain controlled by the child controller, the interface in the message connects the edge node of the upstream domain along the path, the label is allocated for the interface on the node that is the next hop of the edge node.

6.2.2.10. Request for Removing Tunnel Segment in HSCS

When a parent controller receives a request for deleting a tunnel from a user or an application, or receives a reply message for creating tunnel segment with status of Fail from a child controller, the parent controller will delete the tunnel through sending a request message for removing tunnel segment to each of the related child controllers.

A request message for removing tunnel segment (RTSReq message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).

The child controller releases the labels assigned along the path

segments under GTID and PID, and removes the cross connects on each of the nodes along the path segments. If the child controller reserved the link bandwidth along the path segments under GTID and PID, it releases the link bandwidth reserved.

6.2.2.11. Reply for Removing Tunnel Segment in HSCS

After removing the tunnel segment as requested by a request message for removing tunnel segment from a parent controller, a child controller sends the parent controller a reply message for removing tunnel segment.

A reply message for removing tunnel segment (RTSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).
- Status of the tunnel segment removal:

  Success: The tunnel segment requested is removed successfully.

  Fail: The tunnel segment requested can not be removed.

- Error code and reasons for failure if the status is Fail.

6.2.3. Messages for Distributed SDN Control System

Regarding to the operations related to paths and tunnels, most of the messages used in Distributed SDN Control System (DSCS) are the same as their corresponding messages used in Hierarchical SDN Control System (HSCS), which are described in the previous section. Some messages in DSCS are similar to their corresponding messages in HSCS.

6.2.3.1. Message for Connections and Accesses Advertisement in DSCS

After a distributed controller discovers its adjacent distributed controllers, it sends each of them a message for connections and accesses advertisement. The contents of the message is the same as that of the corresponding message that a child controller sends to its parent controller, which is described in section 6.2.2.1.

6.2.3.2. Request for Growing SPT

After receiving a request for creating an end to end tunnel from source A to destination Z for a given set of constraints, a distributed controller D1 allocates a global tunnel identifier (GTID) for the end to end tunnel crossing domains and a path identifier

(PID) for an end to end path to be computed for the tunnel. D1 sends another distributed controller D2 a request message for growing a shortest path tree (SPT). D2 computes a set of path segments in the domain it controls and builds/grows SPT using these path segments as links.

A request message for Growing SPT (GSReq message for short) from a distributed controller D1 to another distributed controller D2 comprises:

- The address or identifier of the start-node (saying X) in the domain controlled by D2. From this node, a number of path segments are to be computed.

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID). For the path of the tunnel identified by the Tunnel-ID-Info, a number of path segments are to be computed.

- The SPT built so far.

- The current candidate-list.

- The constraints for the path such as bandwidth constraints and color constraints.

- A destination node Z. If Z is in the domain controlled by D2, D2 computes a shortest path segment satisfying the constraints from node X to node Z within the domain.

Controller D2 updates the candidate-list using the path segments computed as links, selects a node Y with minimum cost from the candidate-list and adds Y into the SPT.

After adding a node Y into the SPT, D2 constructs a request message for growing SPT using the updated SPT, the updated candidate-list and node Y as a start-node, and sends the message to the distributed controller controlling the domain containing Y.

D2 stores the details about every shortest path segment computed for the path of the tunnel when it sends D3 the request message containing the path segments as links.

D2 deletes the path segments computed for the path of the tunnel after a given period of time if it does not receive any request for keeping them.

6.2.3.3. Reply for Growing SPT

After a distributed controller D2 receives a request message for growing SPT from another distributed controller D1, D2 computes the path segments in the domain it controls and grows SPT using the segments as links. It sends a distributed controller D3 a request message for growing SPT to continue to grow SPT and may send D1 a reply for growing SPT.

A reply message for Growing SPT (GSRep message for short) from D2 to D1 comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID), which identifies the path of the tunnel for which a SPT is being built for.

- Status of growing SPT:

  Success: The SPT is built successfully.

  Fail: The SPT can not be built to contain a shortest path from source A to destination Z.

- SPT if the status is Success.

- Error code and reasons for failure if the status is Fail.

6.2.3.4. Request for Removing Path Segments in DSCS

After a shortest path satisfying a set of constraints from source A to destination Z is computed successfully, the distributed controller D0 originating the path computation may delete the path segments computed and stored in the related distributed controllers, which are not any part of the shortest path. When the path computation fails, D0 may delete the path segments computed and stored in the related distributed controllers. D0 may send a distributed controller Di a request message for removing the path segments computed by Di and stored in Di.

The contents of the request message for removing path segments sent to a distributed controller Di is the same as that of the corresponding message sent to a child controller, which is described in section 6.2.2.4. Di receiving the message performs the same removal operations on the path segments according to the contents of the message as the child controller does.

6.2.3.5. Reply for Removing Path Segments in DSCS

After removing the path segments as requested by a request message for removing path segments from a distributed controller D0, a distributed controller Di sends D0 a reply message for removing path segments.

The contents of the reply message for removing path segments sent by distributed controller Di is the same as that of the corresponding message sent by the child controller, which is described in section 6.2.2.5.

6.2.3.6. Request for Keeping Path Segments in DSCS

After a shortest path satisfying a set of constraints from source A to destination Z is computed, the distributed controller D0 originating the path computation may send a request message for keeping path segments to each of the related distributed controllers Di to keep the path segments on the shortest path.

The contents of the request message for keeping path segments sent to a distributed controller Di is the same as that of the corresponding message sent to a child controller, which is described in section 6.2.2.6. Di receiving the message performs the same retention operations on the path segments according to the contents of the message as the child controller does.

6.2.3.7. Reply for Keeping Path Segments in DSCS

After keeping path segments as requested by a request message for keeping path segments from a distributed controller D0, a distributed controller Di sends D0 a reply message for keeping path segments.

The contents of the reply message for keeping path segments sent by distributed controller Di is the same as that of the corresponding message sent by the child controller, which is described in section 6.2.2.7.

6.2.3.8. Request for Creating Tunnel Segment in DSCS

After obtaining the end to end shortest point to point (P2P) path, a distributed controller creates a tunnel along the path crossing multiple domains through sending a request message for creating tunnel segment to each of the related distributed controllers along the path in a reverse direction to create a tunnel segment.

A request message for creating tunnel segment (CTSReq message for short) sent from controller D0 to Di comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).
- A path segment from a start-point to an end-point.
- A label and an interface if the domain controlled by Di is not a destination domain.

Di allocates and reserves link bandwidth along the path segment identified by the start-point and end-point, assigns labels along the path segment, and writes cross connects on each of the nodes along the path segment.

For the non destination domain, Di writes the cross connect on the edge node to the downstream domain using the label and the interface from the downstream domain in the message.

For the non source domain, Di will include a label and an interface in a message to be sent. The interface connects the edge node of the upstream domain along the path. The label is allocated for the interface on the node that is the next hop of the edge node.

6.2.3.9. Reply for Creating Tunnel Segment in DSCS

After creating tunnel segment as requested by a request message for creating tunnel segment from a distributed controller D0, a distributed controller Di may send D0 a reply message for creating tunnel segment.

A reply message for creating tunnel segment (CTSRep message for short) comprises:

- The Tunnel-ID-Info containing the global tunnel identifier (GTID) and the path identifier (PID).
- Status of the tunnel segment creation:

  Success: The tunnel segment requested is created successfully.

  Fail: The tunnel segments requested can not be created.

- A label and an interface if the domain controlled by Di is not source domain and the status is Success.
- Error code and reasons for failure if the status is Fail.

For the non source domain controlled by Di, the interface in the message connects the edge node of the upstream domain along the path,

the label is allocated for the interface on the node that is the next hop of the edge node.

6.2.3.10. Request for Removing Tunnel Segment in DSCS

When a distriuted controller D0 receives a request for deleting a tunnel from a user or an application, or receives a reply message for creating tunnel segment with status of Fail from another distributed controller, D0 deletes the tunnel through sending a request message for removing tunnel segment to each of the related distributed controllers Di.

The contents of the request message for removing tunnel segment is the same as that of the corresponding message described in section 6.2.2.10. The distributed controller Di receiving the message performs the same removal operations on the tunnel segments according to the contents of the message as the child controller does.

6.2.3.11. Reply for Removing Tunnel Segment in DSCS

After removing the tunnel segment as requested by a request message for removing tunnel segment from a distributed controller D0, a distributed controller Di sends D0 a reply message for removing tunnel segment.

The contents of the reply message for removing tunnel segment sent by distributed controller Di is the same as that of the corresponding message sent by the child controller, which is described in section 6.2.2.11.

6.3. Connections and Accesses Advertisement

6.3.1. Advertisement in Hierarchical SDN Control System

A child controller sends its parent controller a message for connections and accesses, which contains the connections (i.e., inter-domain links) connecting the domain that the child controller controls to its adjacent domains, and the addresses/prefixes (i.e., the access points) in the domain to be accessible from outside of the domain.

When there is a change on the connections and the accesses of the domain, the child controller sends its parent controller a updated message for the connections and accesses, which contains the latest connections and accesses of the domain.

A parent controller stores the connections and accesses for each of its child controllers according to the messages for connections and

accesses received from the child controllers. For a updated message, it updates the connections and accesses accordingly.

When a child controller is down, its parent controller removes the connections and accesses of the domain controlled by the child controller.

After connections and accesses advertisements, a parent controller has the exterior information about all the domains controlled by its child controllers. In other words, a parent controller has the connections among the domains (i.e., the inter-domain links connecting the domains) controlled by its child controllers and the addresses/prefixes (i.e., access points) in the domains to be accessible.

A connection comprises: the attributes for a link connecting domains and the attributes for the end points of the link. The attributes for an end point of a link comprises the type of the end point node such as ABR or ASBR, and the domain of the end point such AS number and area number.

An access point comprises an address or a prefix of a domain to be accessible outside of the domain.

6.3.2. Advertisement in Distributed SDN Control System

Each distributed controller sends its adjacent distributed controllers a message for connections and accesses, which contains the connections (i.e., inter-domain links) from the domain that the controller controls to its adjacent domains, and the addresses (i.e., the access points) in the domain to be accessible from outside of the domain.

When there is a change on the connections and the accesses of the domain, the controller sends its adjacent distributed controllers a updated message for connections and accesses, which contains the latest connections and accesses of the domain.

After a distributed controller A receives a message for connections and accesses originated from a distributed controller X from an adjacent distributed controller B, A stores and/or updates the connections and accesses of the domain controlled by X according to the message for X and sends the message to all its adjacent distributed controllers except for B if the contents of the message for X is changed.

The connections and accesses of the domain controlled by distributed controller X will be removed if X is not reachable.

When a new distributed controller joins through an adjacency, it will get all the connections and accesses messages for the other distributed controllers via the adjacency.

After advertisements of connections and accesses, each distributed controller has the exterior information about all the domains controlled by distributed controllers. In other words, it has the connections among the domains (i.e., the inter-domain links connecting the domains) and the addresses in the domains to be accessible.

6.4. Tunnel Creation

6.4.1. Tunnel Creation in Hierarchical SDN Control System

6.4.1.1. Procedure for Computing Path in HSCS

For a top level parent without domain topology, the parent controller computes a shortest point to point (P2P) path for a tunnel from a source to a destination satisfying a set of constraints given to the tunnel through building a shortest path tree (SPT). The SPT is built from the SPT containing just the source as root and an empty candidate-list in the following steps.

Step 1: The parent controller selects the node X just added to the SPT (Initially, it selects the source).

Step 2: After selecting the node X just added into the SPT, the parent controller chooses the child controller controlling the domain containing the node X, and determines whether the node X is destination.

For destination node, the parent controller stops computing path since the end to end path from source to destination is in the SPT, which is from the root of the SPT to the node (destination node) in the SPT.

For non-destination node X, the parent controller sends the child controller a request message for computing path segments related to the domain controlled by the child controller. The request contains the exception-list for the domain and flag E.

- After receiving the request message, the child controller computes a shortest path segment from node X to each of the edge nodes of the domain not in the exception-list if E is 1.

- In addition, it computes a shortest one hop path segment from node X to each of the edge nodes of the adjacent domains not in the exception-list just using the inter-domain links attached to node X if node X is an edge node and there is an inter-domain link attached to it.

- If node X is in the destination domain, it computes a shortest path segment from node X to the destination.

- It sends the parent controller a reply message with the path segments computed as links.

Step 3: After receiving the reply message from the child controller, the parent controller updates the candidate list with the links, picks up a node in the candidate list with the minimum cost and adds it into the SPT. Repeat step 1.

For a parent without domain topology, if the parent controller is also a child controller of another upper level parent controller, after receiving a request for computing path segments from the upper level parent controller, the parent controller computes each of the path segments as requested in the same way as described above. It records and maintains the path segments computed under the GTID and GPID in the request message received from the upper level parent controller.

In addition, for each path segment to be computed, it allocates a new GTID and GPID for the path segment and computes the path segment through sending a request message for computing path segments to each of its related child controllers using the new GTID and GPID.

When the parent as a child controller receives a request message for removing path segments from the upper level parent controller, it removes the path segments computed by each of its related child controllers through sending a request message for removing path segments to each of the related child controllers, and then it removes the path segments crossing multiple domains controlled by its child controllers.

6.4.1.2. Procedure for Creating Tunnel along Path in HSCS

After obtaining the end to end shortest point to point (P2P) path, the parent controller creates a tunnel along the path crossing multiple domains through requesting the child controllers along the path in a reverse direction.

For a parent without domain topology, the following is the procedure for creating the tunnel along the path, which is initiated by the

parent controller starting from domain X = destination domain.

Step 1: The parent controller sends the child controller controlling domain X a request message for creating tunnel segment in domain X.

- o After receiving the request message from the parent controller, the child controller creates the tunnel segment in domain X it controls through reserving the resources such as link bandwidth, allocating labels along the path segment and writing a cross connect on every node in the domain along the path.

- o If the child controller is not destination controller, the request message contains an label and interface for the next hop of the edge node of domain X. The label is allocated by the controller that controls the downstream domain of domain X. The child controller uses this label and an incoming label allocated for the incoming interface on the edge node to write a cross connect on the edge node.

- o The child controller sends the parent controller a reply message with the status of the tunnel segment creation. The reply message contains an incoming label and interface for the next hop of the edge node of the upstream domain of domain X if domain X is not source domain.

Step 2: The parent controller receives the reply message from child controller C. If the status in the message is Fail, then it removes the tunnel segments created for the tunnel and return with failure for creating the tunnel.

Step 3: If child controller C is the source controller, then the end to end tunnel is created, and the parent controller and the child controllers along the tunnel maintain the information of the tunnel with the GTID and PID. The parent controller returns with success for creating the tunnel.

Step 4: Child controller C is not source controller. The reply message contains the label and interface, the parent controller repeats step 1 with domain X = the upstream domain of domain X. (In other words, it sends a request message to the child controller that controls the domain which is the upstream domain of the domain in which a tunnel segment is just created. The request contains the label and interface.)

For a parent with domain topology, the procedure for creating the tunnel along the path initiated by the parent controller is similar to the one described above, but has a few of changes to it, which are

listed as follows:

- The request message for creating tunnel segment sent to a child controller from the parent controller contains the detailed information about the path segment (such as ERO comprising every hop of the path segment) along which the tunnel segment to be created.

- The child controller does not check or reserve resources such as link bandwidth along the path segment if the parent controller is responsible for allocating and reserving the resources along the path for the tunnel.

- The child controller does not assign any labels along the path segment if the parent controller is responsible for assigning labels along the path for the tunnel. In this case, the request message for creating tunnel segment contains an label for every hop of the path segment. The reply message from the child controller to the parent controller does not contain any label or interface.

When the parent as a child controller receives a request message for creating tunnel segment along a path segment from the upper level parent controller, it gets the path segments for its related child controllers from the path segment in the message.

For the parent with domain topology, it obtains the detailed hop to hop information crossing multiple domains about the path segment stored by the parent controller using the GTID, PID and start-point and end-point of the path segment in the message received. The parent controller creates the tunnel segments in the multiple domains through sending a request message for creating tunnel to each of its related child controllers along the path in a reverse direction.

For the parent without domain topology, it obtains the detailed information about the path segment stored by the parent controller using the GTID, PID and start-point and end-point of the path segment in the message received. The detailed information includes multiple path segments, each of which crosses a domain controlled by one of its related child controllers. These multiple path segments constitute the path segment in the message, which crosses multiple domains. The parent controller creates the tunnel segments in the multiple domains through sending a request message for creating tunnel to each of its related child controllers along the path in a reverse direction. For each of the path segments crossing a domain, the parent controller creates a tunnel segment along the path segment through sending a request message for creating tunnel segment to its child controller controlling the domain.

6.4.2. Tunnel Creation in Distributed SDN Control System

In a Distributed SDN Control System (DSCS), a distributed controller may be designated as a master controller, which receives various requests such as creating an end to end tunnel from users and applications. After the master controller receives a request for creating an end to end tunnel from source A to destination Z for a given set of constraints, it may compute a path for the tunnel first and then create the tunnel along the path through requesting the related distributed controllers. It may compute a path and create the tunnel along the path in one round through requesting the related distributed controllers.

6.4.2.1. Procedure for Computing Path in DSCS

The master controller initiates the computation of an end to end path from source A to destination Z through sending a request message for growing SPT to the source controller. The message contains start-node = A, SPT with just source A and an empty candidate-list. The source controller starts the path computation.

The following is the procedure for computing a path started from Dj controlling domain X (= source domain).

Step 1: Dj controlling domain X receives a request message for growing SPT from controller Di. The message contains start-node A, SPT, candidate-list, destination Z, constrains, GTID and PID.

Step 2: Dj does the following

- If the start-node is the destination, Dj stops computing path since the end to end path from source to destination is in the SPT. The path is from the root of the SPT to the start-node (== destination node) in the SPT.

- Dj computes a shortest path segment from the start-node A to each of the edge nodes of domain X not in the SPT that satisfies the constrains. It also computes a shortest path segment from the start-node A to the destination Z if Z is in domain X. Dj stores the path segments computed under GTID and PID.

- Dj updates the current candidate-list using the path segments as links and the inter-domain links attached to the start-node, selects a node B with the minimum cost from the updated candidate-list if the list is not empty, and adds the node B into the SPT.

- o If the candidate-list is empty, Dj stops computing path since no path can be computed.

- o Dj removes the node B selected from the candidate-list, repeat step 1 through sending a request message for growing SPT to the distributed controller Dk controlling the domain Y containing B. The request message contains the start-node B, SPT with B just added, updated candidate-list, destination Z, constrains, GTID and PID.

6.4.2.2. Procedure for Creating Tunnel along Path in DSCS

There are a couple of ways in which an end to end tunnel is created along the path computed or the tunnel. One way is that the distributed controller originating the path computation initiates and controls the creation of the tunnel. Another way is that the destination controller for the tunnel starts the tunnel creation.

1. Master Controller Controls Tunnel Creation in DSCS

After obtaining the end to end shortest point to point (P2P) path, the distributed controller D0 originating the path computation initiates the creation of a tunnel along the path crossing multiple domains through requesting the related distributed controllers along the path in a reverse direction.

The following is the procedure for creating the tunnel along the path, which is initiated by D0 starting from domain X = destination domain.

Step 1: D0 sends the distributed controller Di controlling domain X a request message for creating tunnel segment in domain X.

- o After receiving the request message from D0, Di creates the tunnel segment in domain X it controls through reserving the resources such as link bandwidth, allocating labels along the path segment and writing a cross connect on every node in the domain along the path.

- o If Di is not destination controller, the request message contains an label and interface for the next hop of the edge node of domain X. The label is allocated by the controller that controls the downstream domain of domain X. Di uses this label and an incoming label allocated for the incoming interface on the edge node to write a cross connect on the edge node.

- o Di sends D0 a reply message with the status of the tunnel segment creation. The reply message contains an incoming label and interface for the next hop of the edge node of the upstream domain of domain X if domain X is not source domain.

Step 2: D0 receives the reply message from Di. If the status in the message is Fail, then it removes the tunnel segments created for the tunnel and return with failure for creating the tunnel.

Step 3: If Di is the source controller, then the end to end tunnel is created, and the controller D0 and each of the related distributed controllers Di along the tunnel maintain the information of the tunnel with the GTID and PID. D0 returns with success for creating the tunnel.

Step 4: Di is not source controller. The reply message contains the label and interface, D0 repeats step 1 with domain X = the upstream domain of domain X. (In other words, it sends a request message to another distributed controller that controls the domain which is the upstream domain of the domain in which a tunnel segment is just created. The request contains the label and interface.)

2. Destination Controller Starts Tunnel Creation

After obtaining the end to end shortest point to point (P2P) path, the distributed controller D0 originating the path computation initiates the creation of a tunnel along the path crossing multiple domains through requesting the destination controller Dn to start the tunnel creation.

The following is the procedure for creating the tunnel along the path started from Dn controlling domain X = destination domain.

Step 1: If Dn controlling domain X receives a request message for creating tunnel segment from controller D0, then

- o Dn creates the tunnel segment in domain X it controls through reserving the resources such as link bandwidth, allocating labels along the path segment and writing a cross connect on every node in the domain along the path segment.

- o If Dn is not destination controller, the request message contains an label and interface for the next hop of the edge node of domain X. The label is allocated by the controller that controls the downstream domain of domain X. Dn uses this label and an incoming label allocated for the incoming interface on the edge node to write a cross connect on the edge node.

- If Dn is not source controller, repeat step 1 by sending distributed controller Dm controlling the upstream domain W of domain X a request message for creating tunnel segment in domain W if creating tunnel segment in domain X is successful. The request message contains an incoming label and interface for the next hop of the edge node of domain W.

- If creating tunnel segment in domain X fails, Dn removes the partial tunnel segment created in domain X and repeat step 1 through sending its downstream controller D0 a reply message with the status of Fail.

- If Dn is the source controller, repeat step 1 through sending its downstream controller D0 a reply message with the status of Success.

Step 2: If Dn receives a reply message from its upstream controller Dm, then

- If the status in the message is Fail, Dn removes the tunnel segments created for the tunnel and sends its downstream controller D0 a reply message with the status of Fail.

- If the status in the message is Success, Dn sends D0 a reply message with the status of Success.

6.4.2.3. Computing Path and Creating Tunnel in One Round

The master controller initiates the computation of an end to end path from source A to destination Z. The source controller starts computing the path through growing the SPT from source as root. Other related controllers continue computing the path through growing the SPT. The destination controller finishes computing the path and gets the path when the destination Z is in SPT.

After the destination controller gets the path, it tells the master controller that the path is computed through a reply message for growing SPT and starts creating the tunnel.

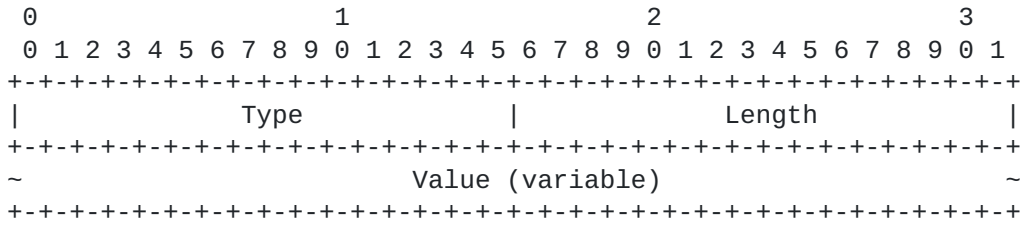
It creates the tunnel segment in the destination domain and then sends the controller Du controlling the upstream domain U of the destination domain a request message for creating tunnel segment along the path. Du creates the tunnel segment in domain U and then sends the controller controlling the upstream domain of domain U a request message for creating tunnel segment along the path, and so on. At last, the source controller finishes creating the tunnel after it creates the tunnel segment in the source domain.

The source controller tells the master controller that the tunnel is created successfully after it finishes creating the tunnel through a reply message for creating tunnel segment.

6.5. Transporting SDN Information in BGP

6.5.1. TLV Format

The information in the new SDN NLRIs is encoded in Type/Length/Value triplets. The TLV format is shown below.



The Length field defines the length of the value portion in octets (thus a TLV with no value portion would have a length of zero). The TLV is not padded to four-octet alignment. Unrecognized types MUST be preserved and propagated. In order to compare NLRIs with unknown TLVs all TLVs MUST be ordered in ascending order by TLV Type. If there are more TLVs of the same type, then the TLVs MUST be ordered in ascending order of the TLV value within the TLVs with the same type by treating the entire value field an opaque hexadecimal string and comparing leftmost octets first regardless of the length of the string. All TLVs that are not specified as mandatory are considered optional.

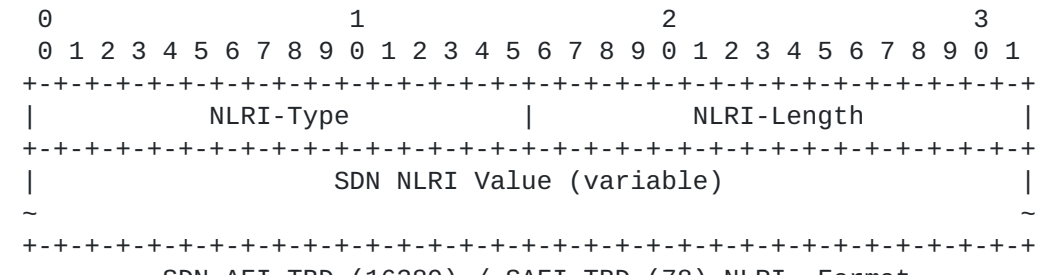
6.5.2. SDN NLRIs

The MP_REACH_NLRI and MP_UNREACH_NLRI attributes are BGP's containers for carrying opaque information. Each SDN NLRI describes a SDN message.

All SDN messages SHALL be encoded using AFI TBD (16389) / SAFI TBD (78).

In order for two BGP speakers to exchange SDN NLRI, they MUST use BGP Capabilities Advertisement to ensure that they both are capable of properly processing such NLRI. This is done as specified in [RFC4760], by using capability code 1 (multi-protocol BGP), with AFI TBD (16389) / SAFI TBD (78) for BGP-SDN

The format of the SDN NLRI is shown as follows.

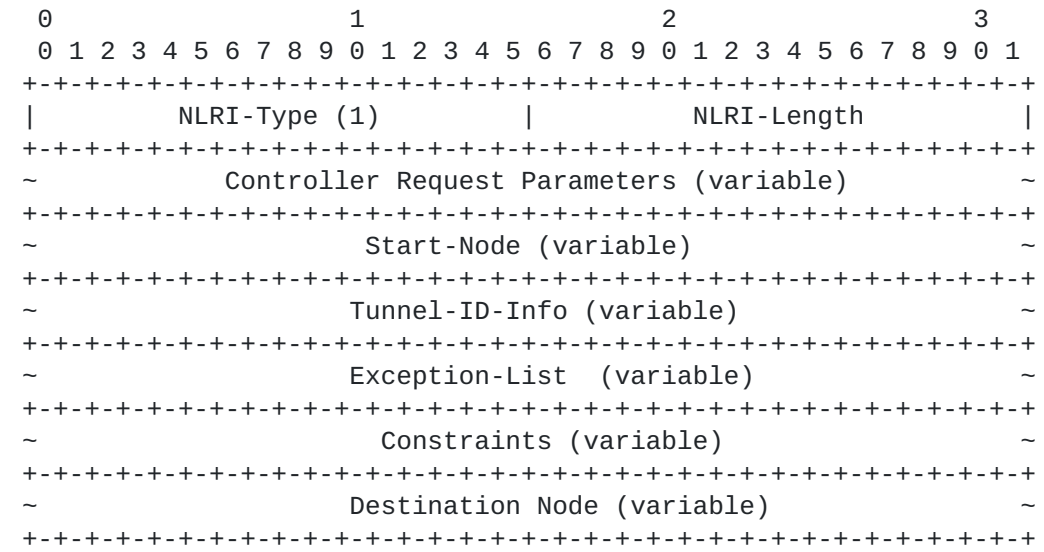


SDN AFI TBD (16389) / SAFI TBD (78) NLRI Format

Where the values of NLRI-Type are defined below.

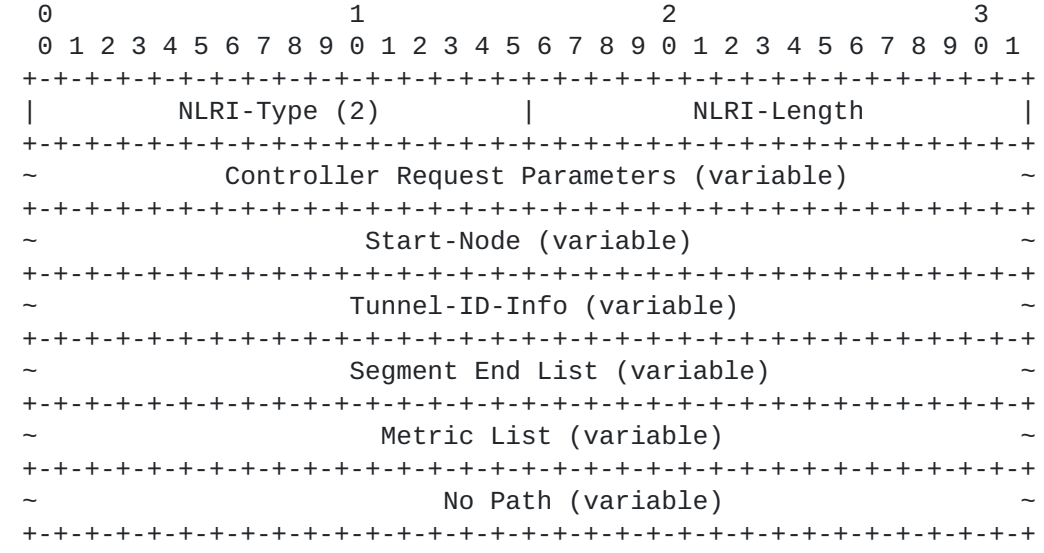
Value of NLRI-Type	Meaning
1	Request for Computing Path Segment (CPSReq)
2	Reply for Computing Path Segment (CPSRep)
3	Request for Removing Path Segment (RPSReq)
4	Reply for Removing Path Segment (RPSRep)
5	Request for Keeping Path Segment (KPSReq)
6	Reply for Keeping Path Segment (KPSRep)
7	Request for Creating Tunnel Segment (CTSReq)
8	Reply for Creating Tunnel Segment (CTSRep)
9	Request for Removing Tunnel Segment (RTSReq)
10	Reply for Removing Tunnel Segment (RTSRep)
11	Message for Connection and Access Advertisement(CAA)
12	Request for Growing SPT (GSReq)
13	Reply for Growing SPT (GSRep)

The format of Request for Computing Path Segment NLRI (NLRI-Type = 1) is illustrated below.



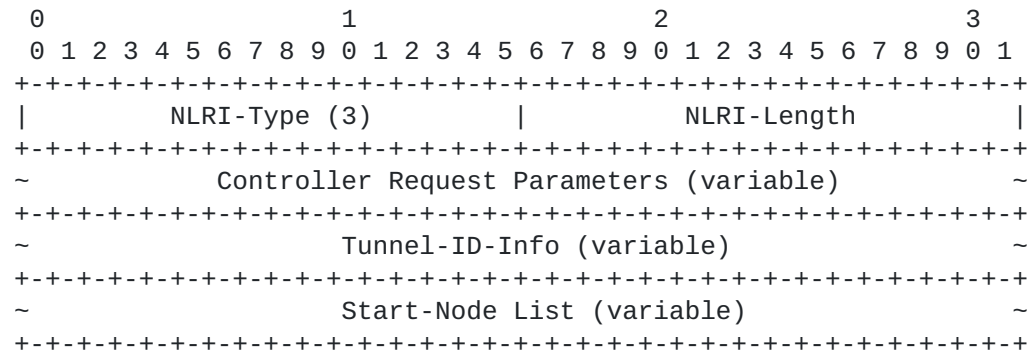
Format of Request for Computing Path Segment NLRI

The format of Reply for Computing Path Segment NLRI (NLRI-Type = 2) is illustrated below.



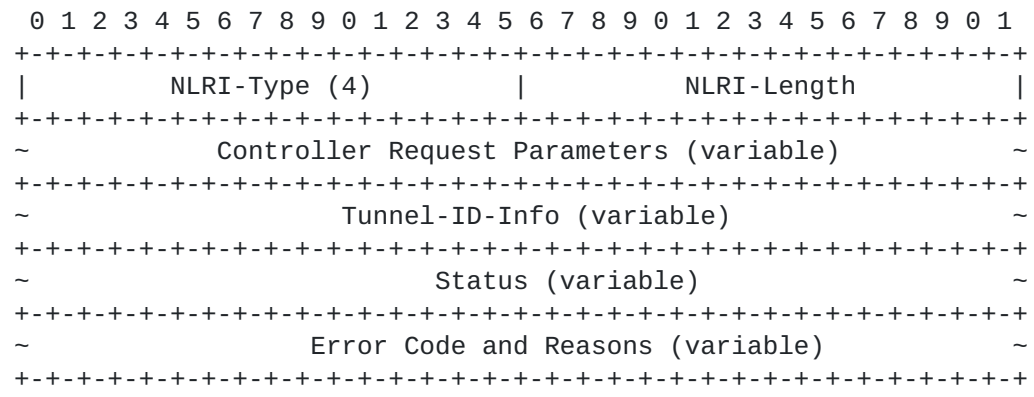
Format of Reply for Computing Path Segment NLRI

The format of Request for Removing Path Segment NLRI (NLRI-Type = 3) is illustrated below.



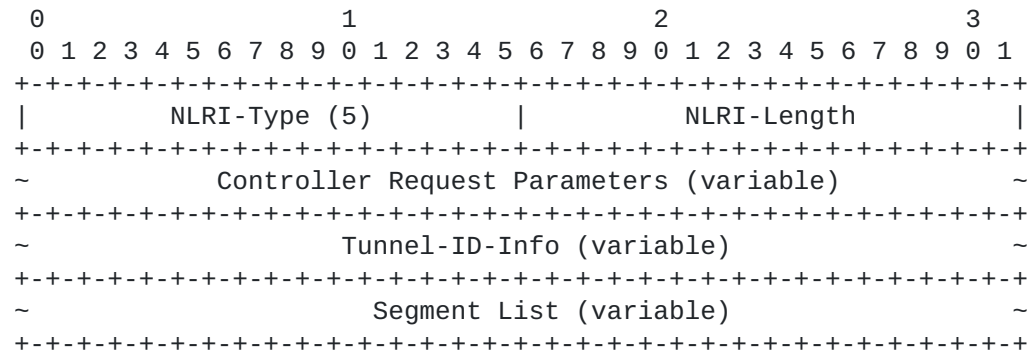
Format of Request for Removing Path Segment NLRI

The format of Reply for Removing Path Segment NLRI (NLRI-Type = 4) is illustrated below.



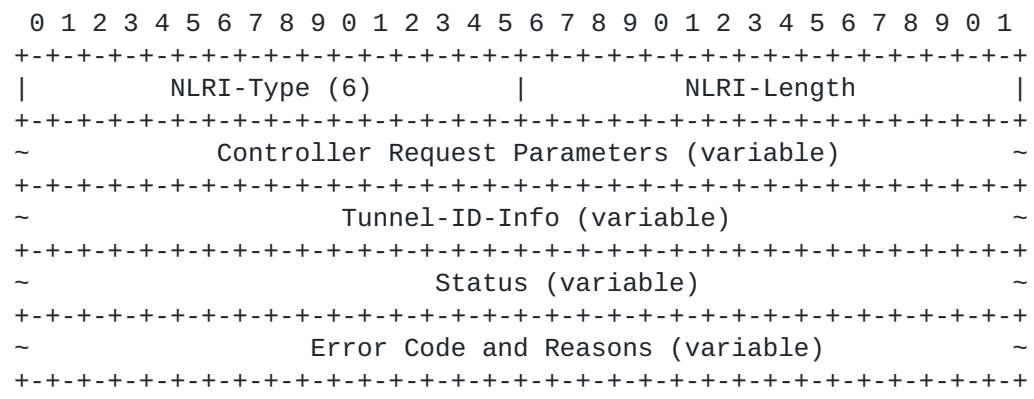
Format of Reply for Removing Path Segment NLRI

The format of Request for Keeping Path Segment NLRI (NLRI-Type = 5) is illustrated below.



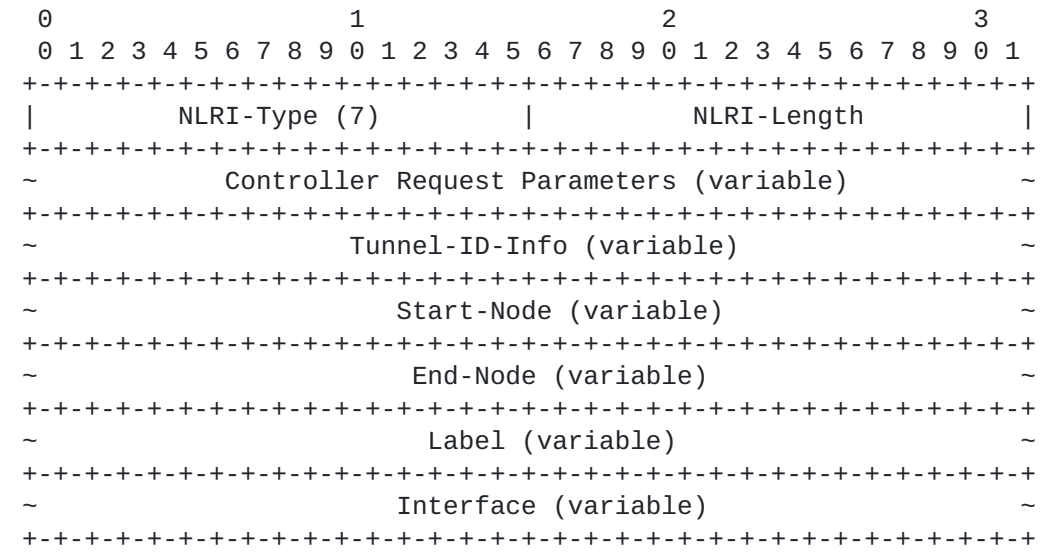
Format of Request for Keeping Path Segment NLRI

The format of Reply for Keeping Path Segment NLRI (NLRI-Type = 6) is illustrated below.



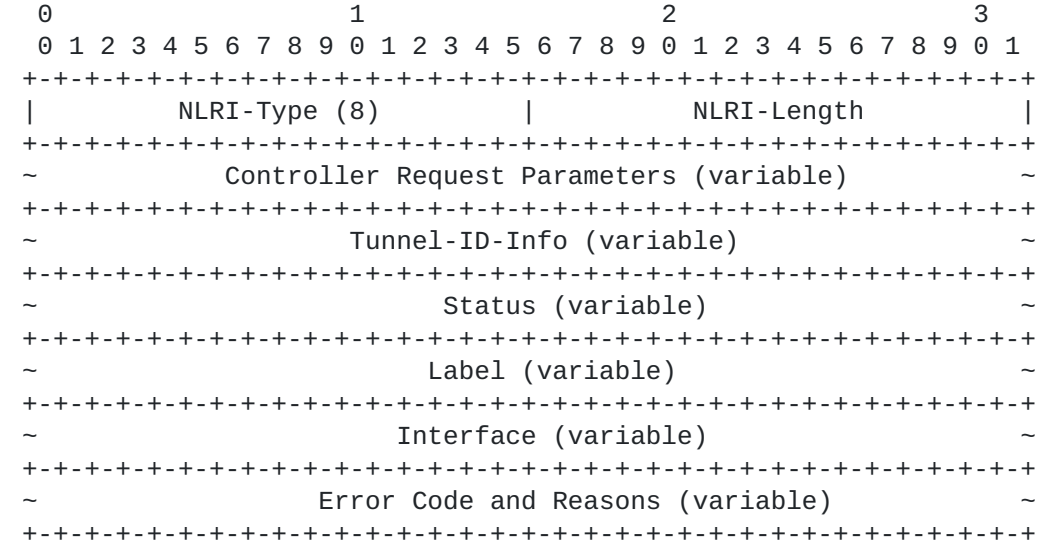
Format of Reply for Keeping Path Segment NLRI

The format of Request for Creating Path Segment NLRI (NLRI-Type = 7) is illustrated below.



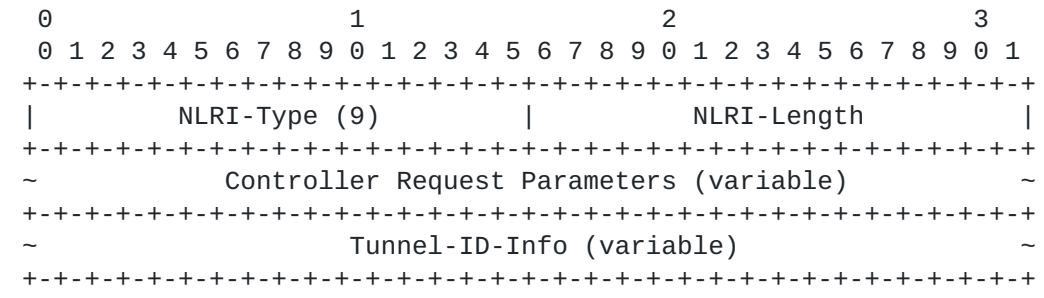
Format of Request for Creating Path Segment NLRI

The format of Reply for Creating Path Segment NLRI (NLRI-Type = 8) is illustrated below.



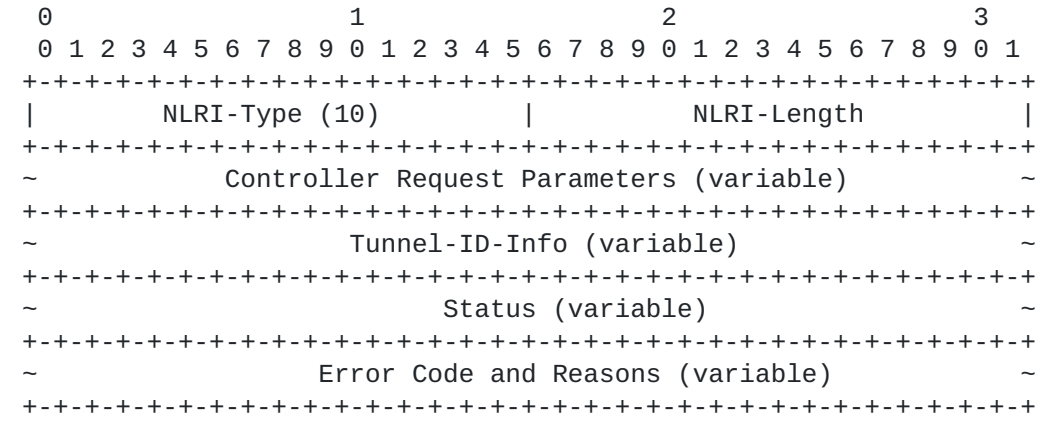
Format of Reply for Creating Path Segment NLRI

The format of Request for Removing Tunnel Segment NLRI (NLRI-Type = 9) is illustrated below.



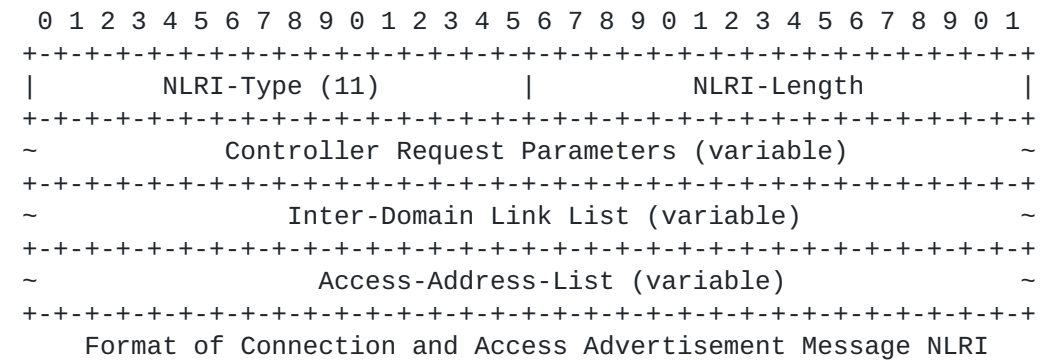
Format of Request for Removing Tunnel Segment NLRI

The format of Reply for Removing Tunnel Segment NLRI (NLRI-Type = 10) is illustrated below.



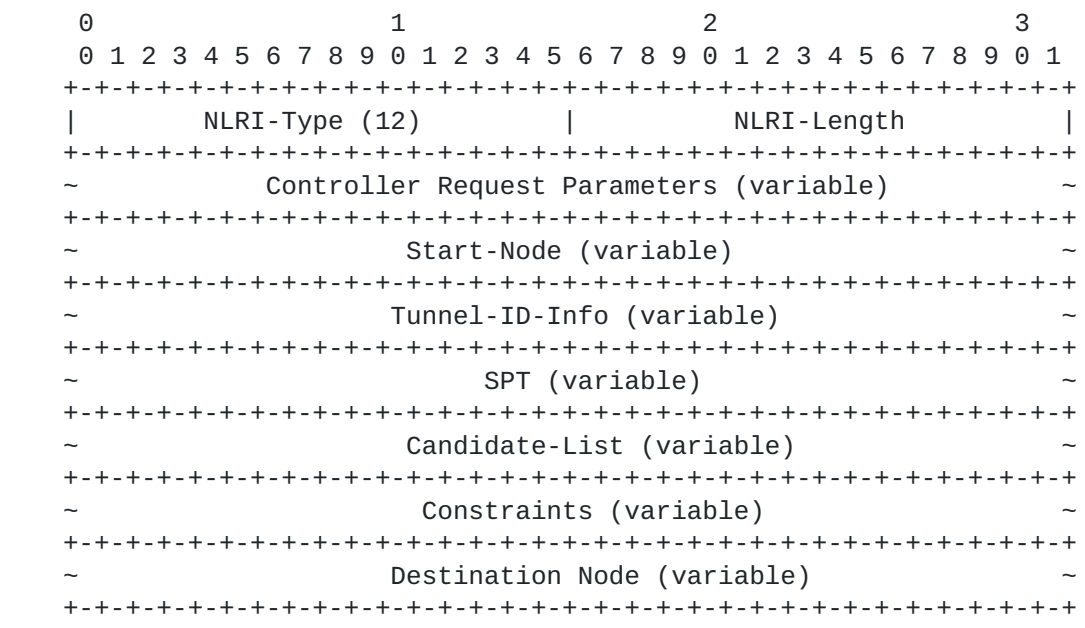
Format of Reply for Removing Tunnel Segment NLRI

The format of Connection and Access Advertisement Message NLRI (NLRI-Type = 11) is illustrated below.



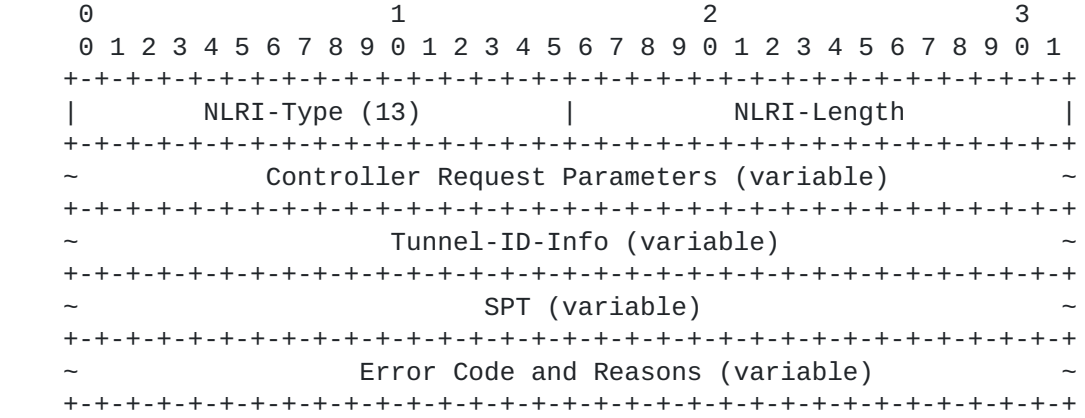
Format of Connection and Access Advertisement Message NLRI

The format of Request for Growing SPT NLRI (NLRI-Type = 12) is illustrated below.



Format of Request for Growing SPT NLRI

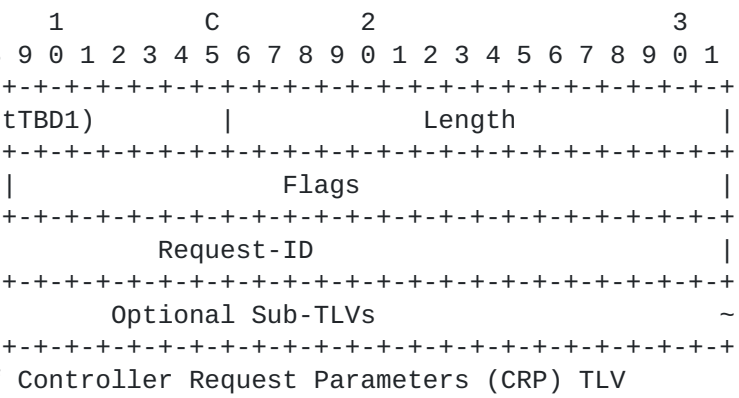
The format of Reply for Growing SPT NLRI (NLRI-Type = 13) is illustrated below.



Format of Reply for Growing SPT NLRI

6.5.3. Controller Request Parameters TLV

A Controller Request Parameters (CRP) TLV carried within each of the new messages for supporting HSCS is used to specify various parameters of a tunnel related operation request. The format of CRP TLV is as follows



Format of Controller Request Parameters (CRP) TLV

The OP (Optimization Parameter) of 4 bits in Flags field are defined as follows:

OP	Meaning (Optimize on)
1	IGP metric
2	TE metric
3	Hop count
4 - 15,0	Undefined/Reserved

The Optimization Parameter (OP) is common for HSCS, DSCS and other control systems.

The following flags are currently defined for HSCS:

- E (Edges of Domain): E set to 1 indicating computing a shortest path segment satisfying a given set of constraints from a start node to each of the edge nodes of the domain controlled by a child controller except for the nodes in a given exception-list.

The PTP (Path computation and Tunnel creation Procedure) of 4 bits in Flags field for DSCS are defined as follows:

PTP	Meaning(Procedure indicated)
1	Master controller Controls Path computation (MCP)
2	Master controller Controls Tunnel creation (MCT)
3	Master controller Initiates Path computation (MIP)
4	Master controller Initiates Tunnel creation (MIT)
5	Master controller Initiates Path computation and Tunnel creation in one round (MIPT)
6 - 15,0	Undefined/Reserved

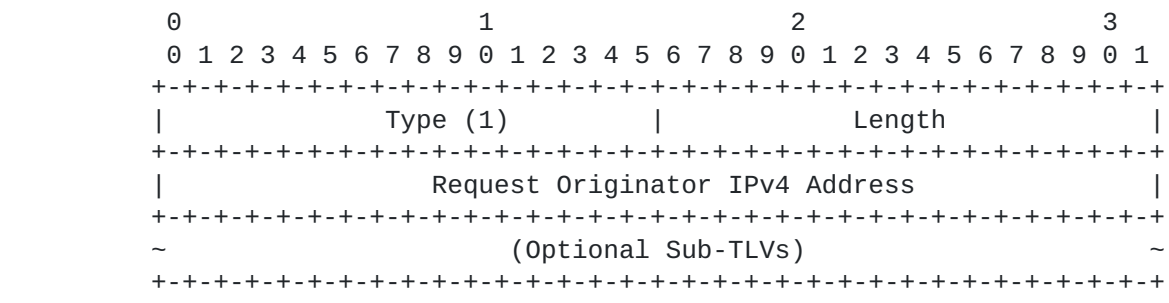
The following optional Sub-TLVs are currently defined for DSCS:

- Request Originator: A distributed controller that originates the request. For example, the master controller is the request originator for the request initiating an end to end path computation when it sends the request to the source controller(, which is the request target controller).

- Request From Controller: A distributed controller that constructs the request and from which the request is sent. For example, in step 2 of the procedure for computing path in DSCS, when distributed controller Dj sends a request message for growing SPT to the distributed controller Dk controlling the domain Y containing B, Dj is the Request From Controller and Dk is the Request Target Controller.

- Request Target Controller: A distributed controller to which the request targets.

The Request Originator Sub-TLV for IPv4 (RO-IPv4 for short) has the following format:



Format of IPv4 Request Originator Sub-TLV

The RO-IPv4 has a 32-bit Controller IPv4 Address for the request

originator. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

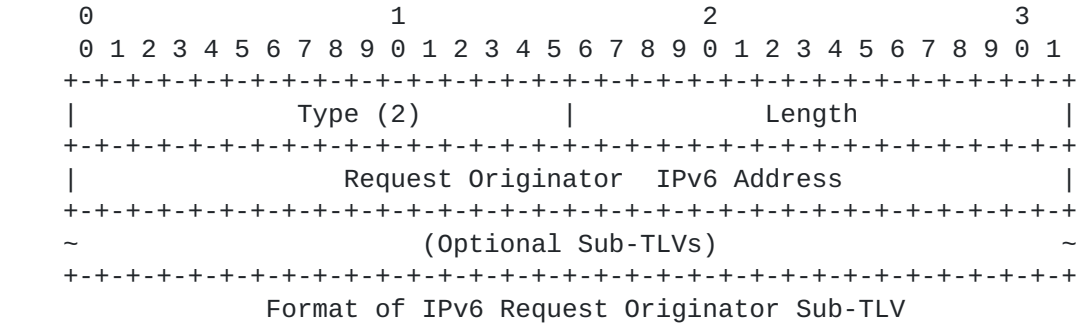
A request originator is a master controller normally in a DSCS. If the master controller is not adjacent to the target controller, to which the master controller sends a request message, it sends the message to its adjacent distributed controller which is on the route to the target controller. The adjacent distributed controller is the next hop node to the target controller according to the routing table.

Each of the distributed controllers along the route to the target controller of the message relays the message to its adjacent distributed controller as the next hop node to the target controller after receiving the message. When the target controller receives the message, it performs the operations as requested by the message.

When a distributed controller such as a target controller sends a reply message to the request originator such as the master controller, it sends the message to its adjacent distributed controller which is on the route to the request originator if the originator is not its adjacent controller.

Each of the distributed controllers along the route to the originator such as the master controller relays the reply message to its adjacent distributed controller as the next hop node to the originator after receiving the reply message. When the originator receives the reply message, it performs the operations according to the message.

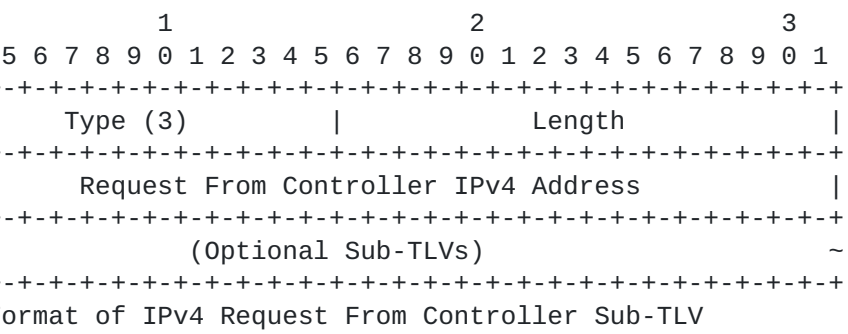
The Request Originator Sub-TLV for IPv6 (RO-IPv6 for short) has the following format:



Format of IPv6 Request Originator Sub-TLV

The RO-IPv6 has a 128-bit Controller IPv6 Address for the request originator. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

The Request From Controller Sub-TLV for IPv4 (RFC-IPv4 for short) has the following format:

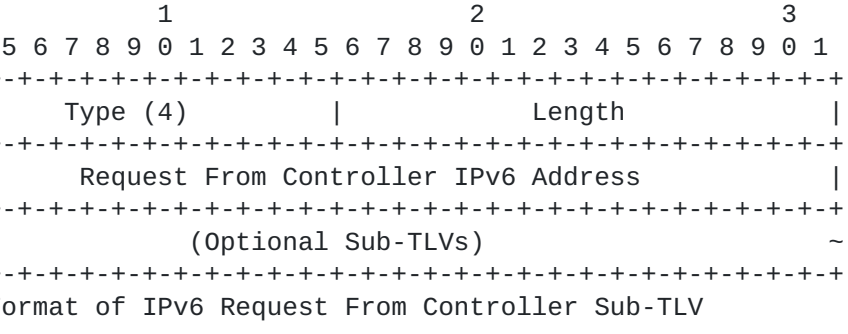


Format of IPv4 Request From Controller Sub-TLV

The RFC-IPv4 has a 32-bit IPv4 Address of the Request From Controller. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

If the Request From controller is not adjacent to the target controller, to which the Request From controller sends a request message, it sends the request message to its adjacent distributed controller which is on the route to the target controller. The adjacent distributed controller is the next hop node to the target controller according to the routing table.

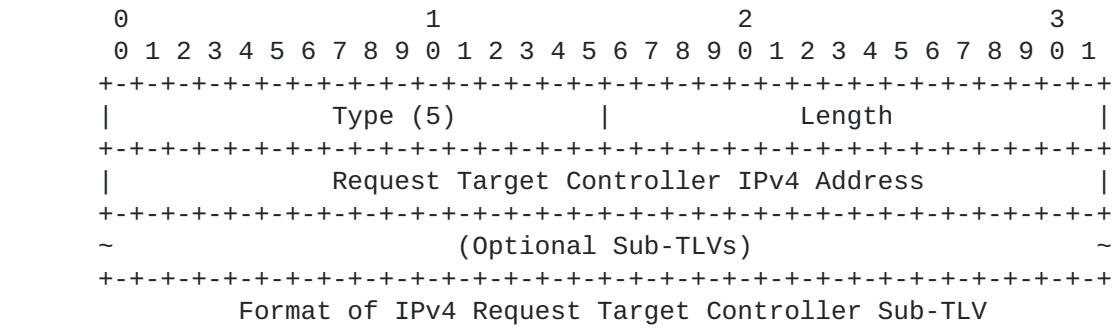
The Request From Controller Sub-TLV for IPv6 (RFC-IPv6 for short) has the following format:



Format of IPv6 Request From Controller Sub-TLV

The RFC-IPv6 has a 128-bit IPv6 Address of the Request From Controller. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

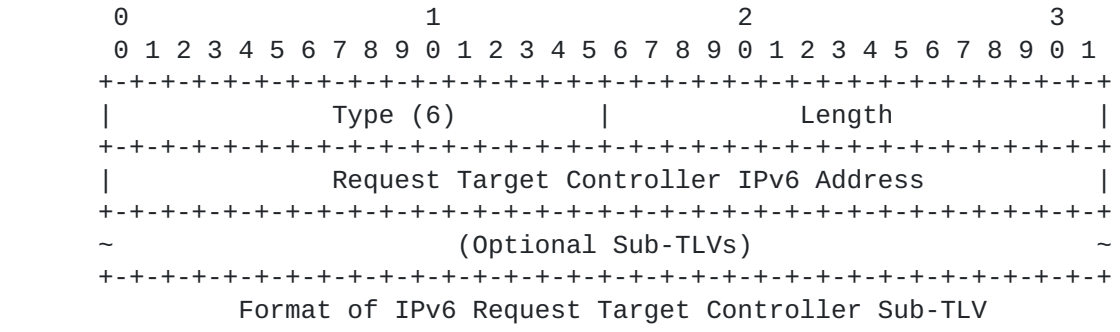
The Request Target Controller Sub-TLV for IPv4 (RTC-IPv4 for short) has the following format:



Format of IPv4 Request Target Controller Sub-TLV

The RTC-IPv4 has a 32-bit IPv4 Address of the Request Target Controller. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

The Request Target Controller Sub-TLV for IPv6 (RTC-IPv6 for short) has the following format:



Format of IPv6 Request Target Controller Sub-TLV

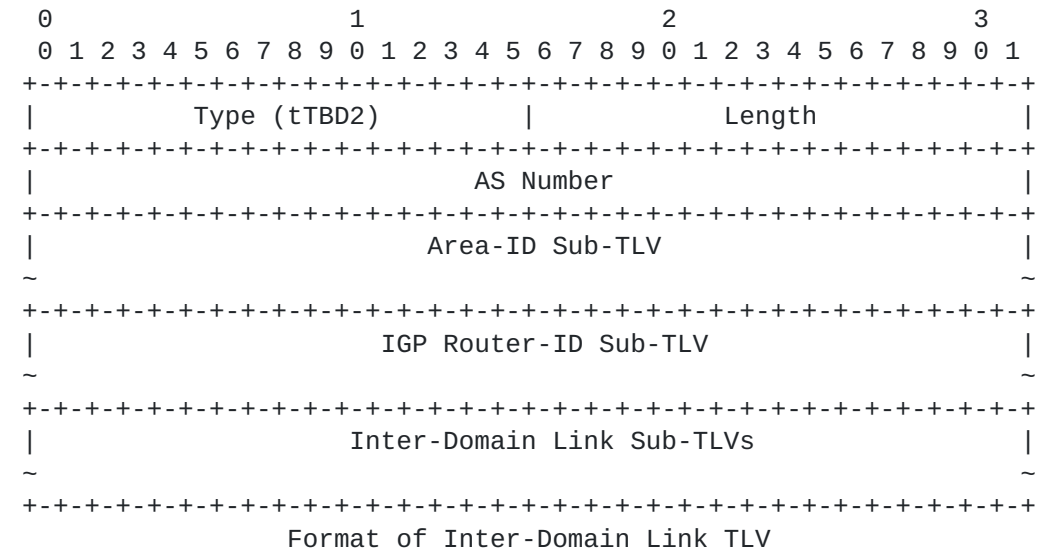
The RTC-IPv6 has a 128-bit IPv6 Address of the Request Target Controller. It may contain additional Sub-TLVs. No Sub-TLVs are currently defined.

6.5.4. CONNECTION and ACCESS TLVs

Three TLVs for CONNECTION and ACCESS are defined:

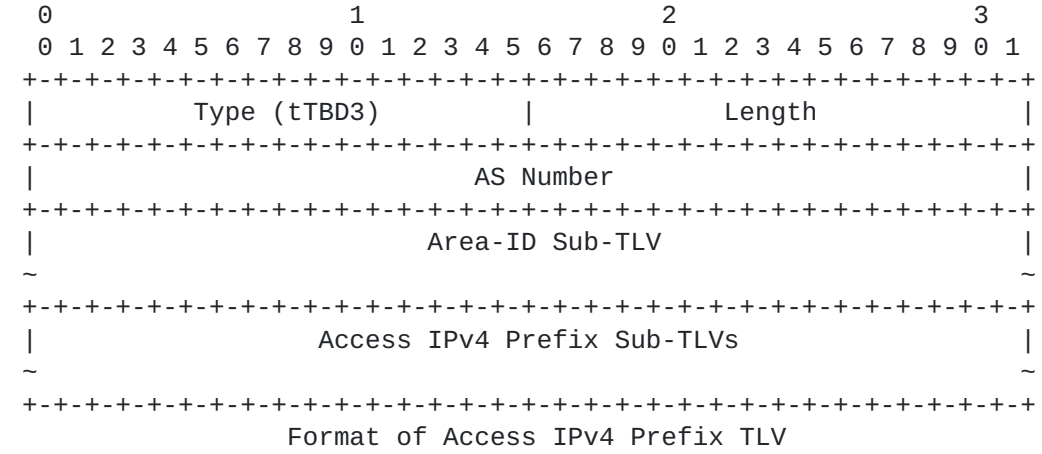
- Inter-Domain Link TLV
- Access IPv4 Prefix TLV
- Access IPv6 Prefix TLV

The format of each of these TLVs is as follows:

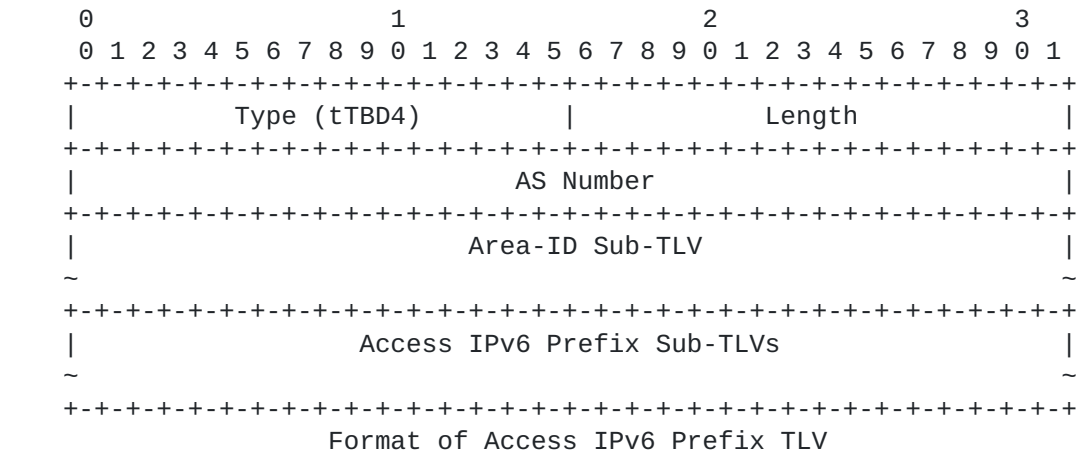


Format of Inter-Domain Link TLV

The Inter-Domain Link TLV describes an inter-domain link and comprises a number of inter-domain link Sub-TLVs.

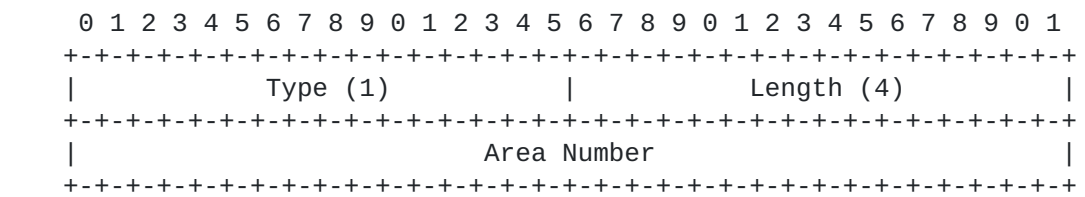


Format of Access IPv4 Prefix TLV



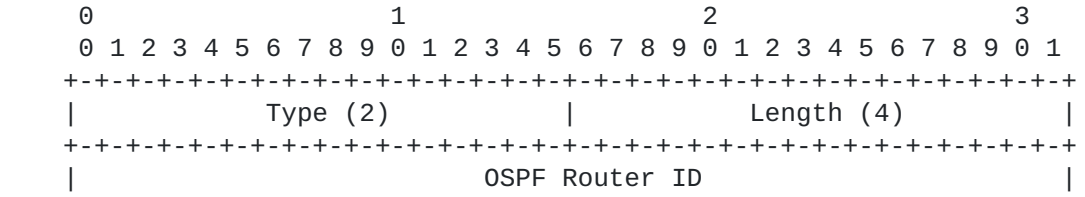
Format of Access IPv6 Prefix TLV

The format of the Area-ID Sub-TLV is shown below:



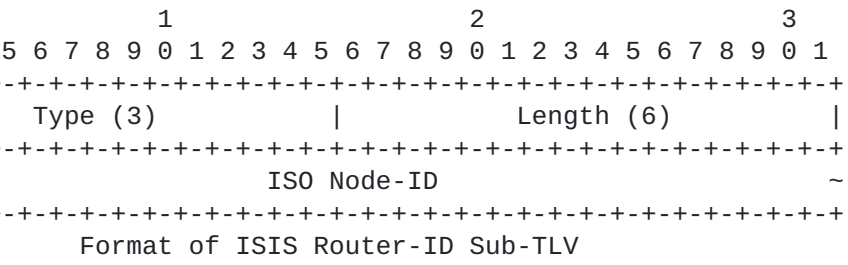
Format of Area-ID Sub-TLV

The format of the OSPF Router-ID Sub-TLV is shown below:



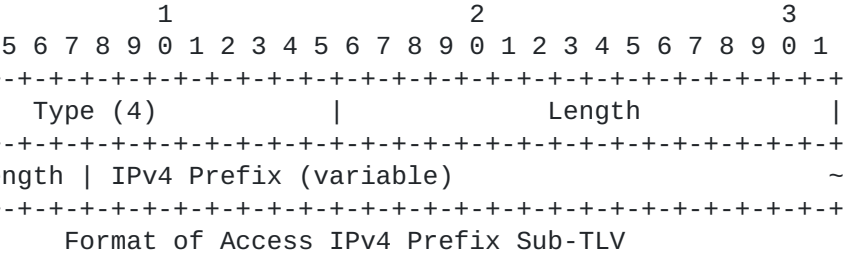
Format of OSPF Router-ID Sub-TLV

The format of the ISIS Router-ID Sub-TLV is shown below:



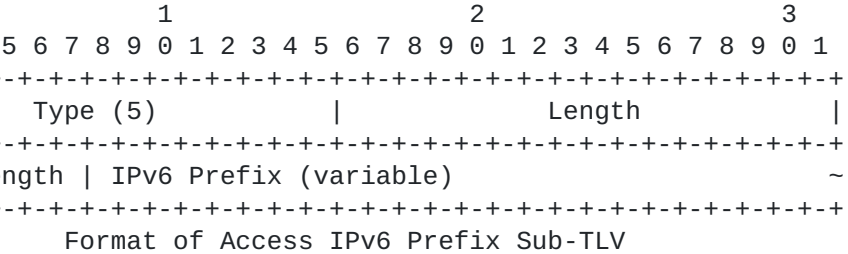
Format of ISIS Router-ID Sub-TLV

The format of the Access IPv4 Prefix Sub-TLV is shown as follows:



Format of Access IPv4 Prefix Sub-TLV

The format of the Access IPv6 Prefix Sub-TLV is illustrated below:



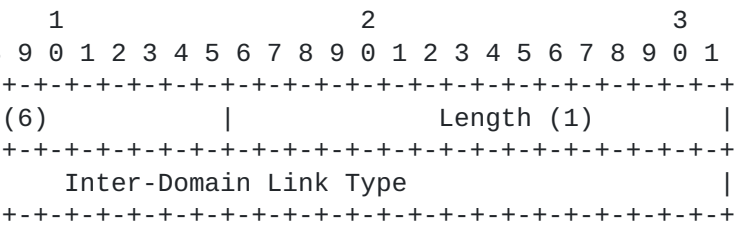
Format of Access IPv6 Prefix Sub-TLV

A number of Inter-Domain link Sub-TLVs are defined:

- Inter-Domain Link Type Sub-TLV
- Remote AS Number ID Sub-TLV
- Remote Area-ID Sub-TLV
- Remote OSPF Router-ID Sub-TLV
- Remote ISIS Router-ID Sub-TLV

- IPv4 Remote ASBR ID Sub-TLV
- IPv6 Remote ASBR ID Sub-TLV
- Local Interface IPv4 Address Sub-TLV
- Local Interface IPv6 Address Sub-TLV
- Remote Interface IPv4 Address Sub-TLV
- Remote Interface IPv6 Address Sub-TLV

The format of the Inter-Domain Link Type Sub-TLV is illustrated below:



Format of Inter-Domain Link Type Sub-TLV

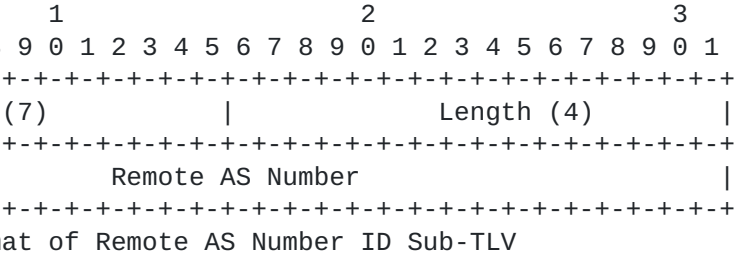
The Inter-Domain Link Type sub-TLV defines the type of the inter-domain link:

1 - Point-to-point

2 - Multi-access

The Inter-Domain Link Type sub-TLV is TLV type 1, and is one octet in length.

The format of the Remote AS Number ID Sub-TLV is illustrated below:

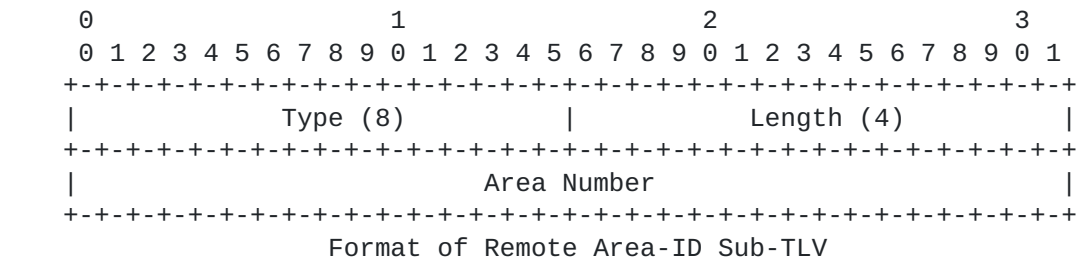


Format of Remote AS Number ID Sub-TLV

The Remote AS Number field has 4 octets. When only two octets are

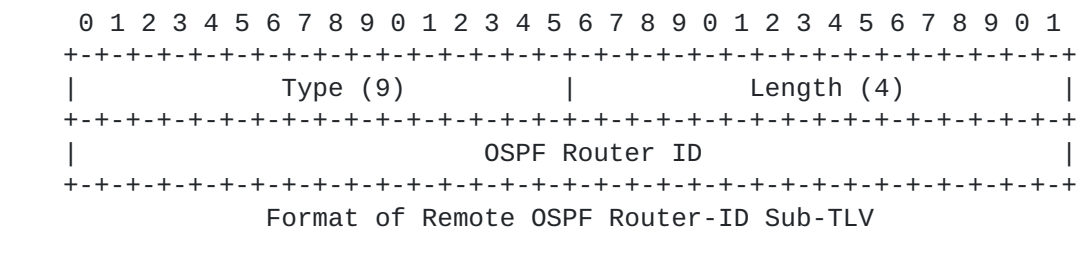
used for the AS number, as in current deployments, the left (high-order) two octets MUST be set to zero.

The format of the Remote Area-ID Sub-TLV is shown below:



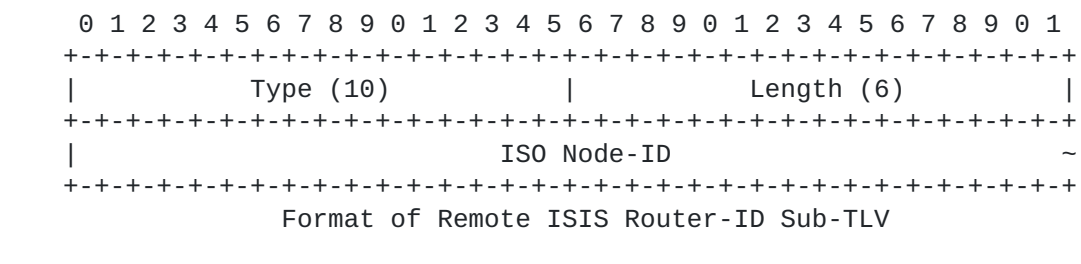
Format of Remote Area-ID Sub-TLV

The format of the Remote OSPF Router-ID Sub-TLV is shown below:



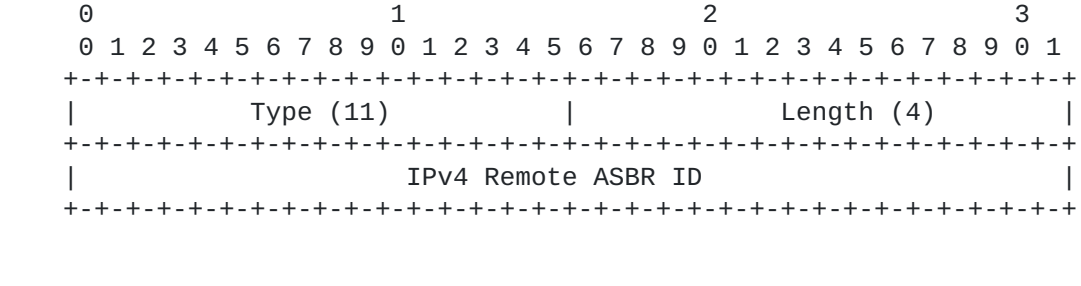
Format of Remote OSPF Router-ID Sub-TLV

The format of the Remote ISIS Router-ID Sub-TLV is shown below:



Format of Remote ISIS Router-ID Sub-TLV

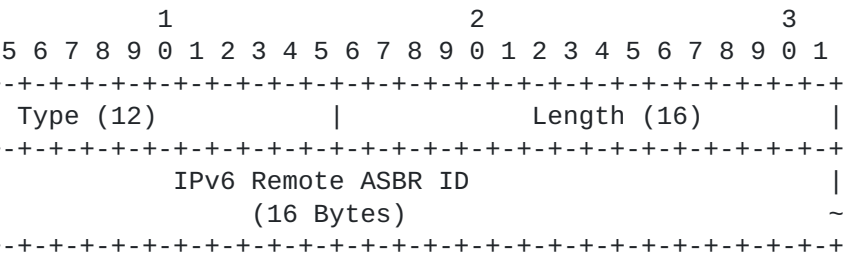
The format of the IPv4 Remote ASBR ID Sub-TLV is illustrated below:



Format of IPv4 Remote ASBR ID Sub-TLV

The IPv4 Remote ASBR ID sub-TLV MUST be included if the neighboring ASBR has an IPv4 address.

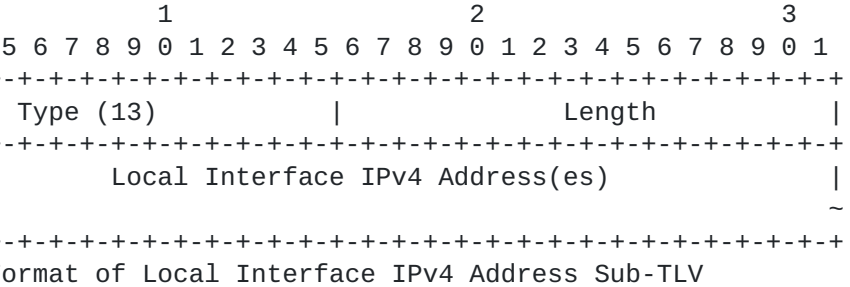
The format of the IPv6 Remote ASBR ID Sub-TLV is illustrated below:



Format of IPv6 Remote ASBR ID Sub-TLV

The IPv6 Remote ASBR ID sub-TLV MUST be included if the neighboring ASBR has an IPv6 address.

The format of the Local Interface IPv4 Address Sub-TLV is shown below:

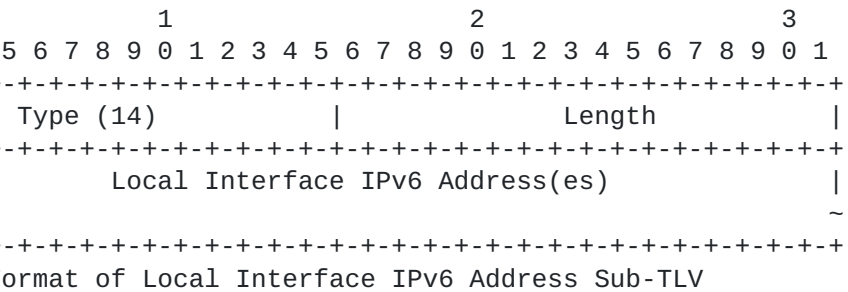


Format of Local Interface IPv4 Address Sub-TLV

The Local Interface IPv4 Address sub-TLV specifies the IPv4 address(es) of the interface corresponding to the inter-domain link. If there are multiple local addresses on the link, they are all listed in this sub-TLV.

The Local Interface IPv4 Address sub-TLV is TLV type 8, and is 4N octets in length, where N is the number of local IPv4 addresses.

The format of the Local Interface IPv6 Address Sub-TLV is illustrated below:

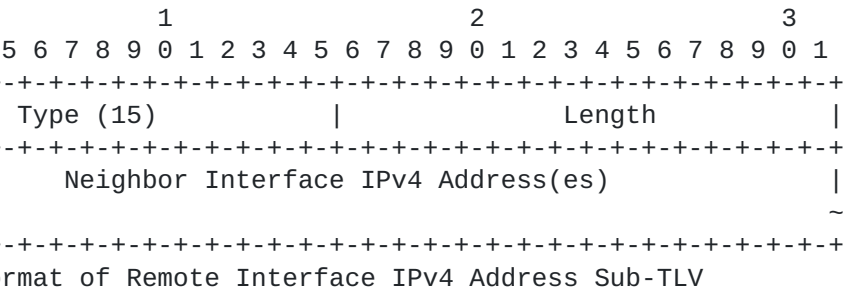


Format of Local Interface IPv6 Address Sub-TLV

The Local Interface IPv6 Address sub-TLV specifies the IPv6 address(es) of the interface corresponding to the inter-domain link. If there are multiple local addresses on the link, they are all listed in this sub-TLV.

The Local Interface IPv6 Address sub-TLV is TLV type 9, and is 16N octets in length, where N is the number of local IPv6 addresses.

The format of the Remote Interface IPv4 Address Sub-TLV is illustrated below:

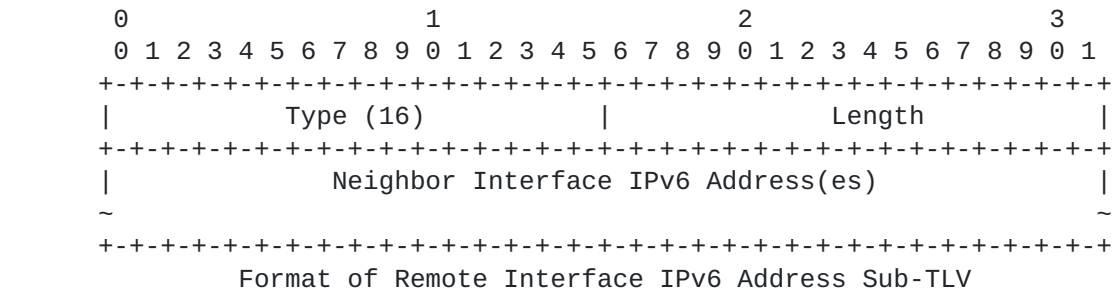


Format of Remote Interface IPv4 Address Sub-TLV

The Remote Interface IPv4 Address sub-TLV specifies the IPv4 address(es) of the neighbor's interface corresponding to the inter-domain link. This and the local address are used to discern multiple parallel links between systems. If there are multiple remote addresses on the link, they are all listed in this sub-TLV.

The Remote Interface IPv4 Address sub-TLV is TLV type 10, and is 4N octets in length, where N is the number of neighbor IPv4 addresses.

The format of the Remote Interface IPv6 Address Sub-TLV is illustrated below:



Format of Remote Interface IPv6 Address Sub-TLV

The Remote Interface IPv6 Address sub-TLV specifies the IPv6 address(es) of the neighbor's interface corresponding to the inter-domain link. If there are multiple neighbor addresses on the link, they are all listed in this sub-TLV.

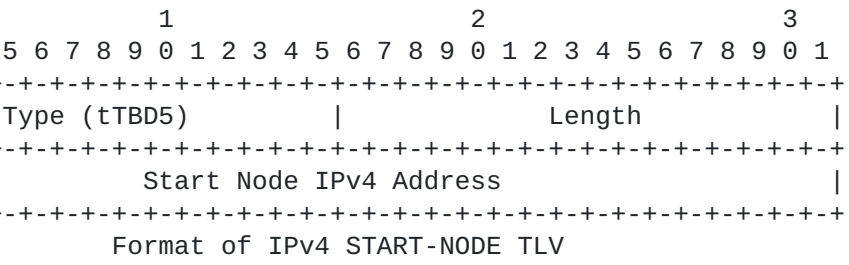
The Remote Interface IPv6 Address sub-TLV is TLV type 11, and is 16N octets in length, where N is the number of neighbor IPv6 addresses.

6.5.5. NODE TLVs

A number of Node TLVs are defined below:

1. IPv4 START-NODE TLV
2. IPv6 START-NODE TLV
3. IPv4 DESTINATION-NODE-LIST TLV
4. IPv6 DESTINATION-NODE-LIST TLV
5. IPv4 SEGMENT-END-NODE-LIST TLV
6. IPv6 SEGMENT-END-NODE-LIST TLV
7. IPv4 EXCEPTION-LIST TLV
8. IPv6 EXCEPTION-LIST TLV
9. IPv4 CANDIDATE-LIST TLV
10. IPv6 CANDIDATE-LIST TLV
11. NODE-COST-LIST TLV

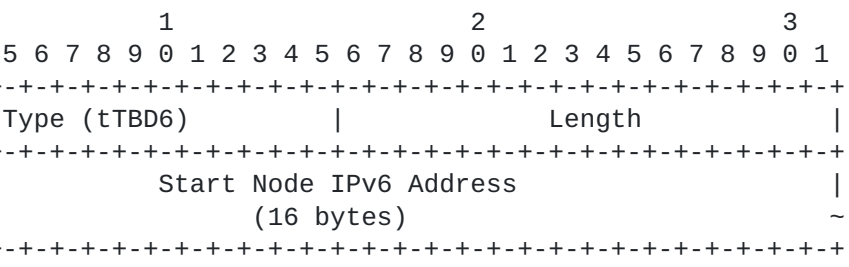
The format of IPv4 START-NODE TLV is as follows:



Format of IPv4 START-NODE TLV

The Start Node IPv4 Address is the IPv4 address of a start node.

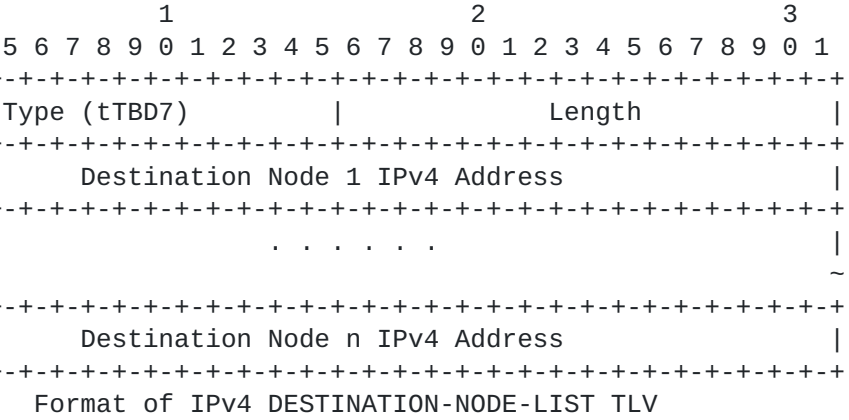
The format of IPv6 START-NODE TLV is as follows:



Format of IPv6 START-NODE TLV

The Start Node IPv6 Address is the IPv6 address of a start node.

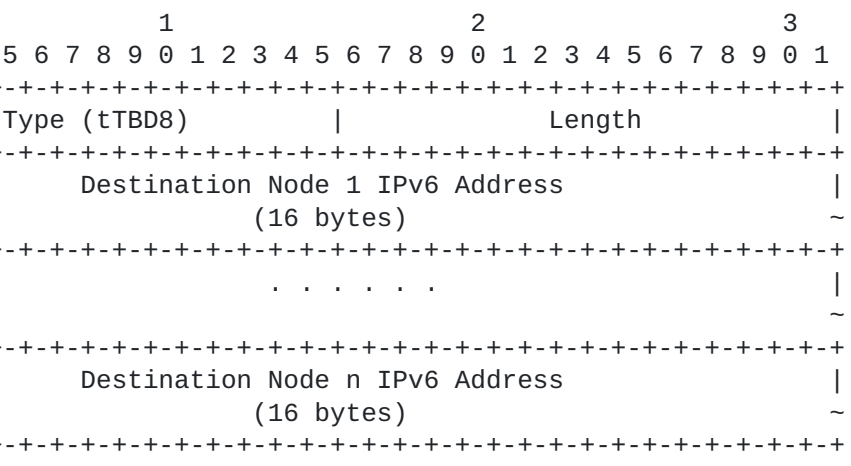
The format of IPv4 DESTINATION-NODE-LIST TLV is as follows:



Format of IPv4 DESTINATION-NODE-LIST TLV

The IPv4 DESTINATION-NODE-LIST contains n destination node IPv4 addresses. An IPv4 DESTINATION-NODE-LIST is also called an IPv4 DESTINATION-NODES.

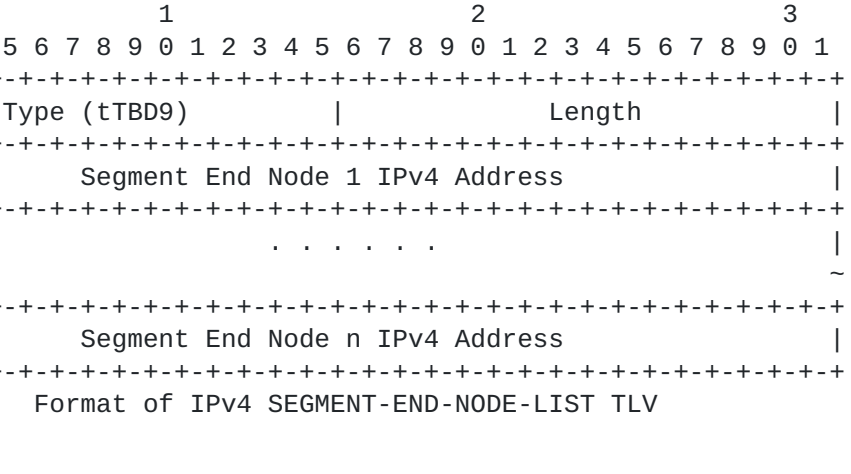
The format of IPv6 DESTINATION-NODE-LIST TLV is as follows:



Format of IPv6 DESTINATION-NODE-LIST TLV

The IPv6 DESTINATION-NODE-LIST contains n destination node IPv6 addresses. An IPv6 DESTINATION-NODE-LIST is also called an IPv6 DESTINATION-NODES.

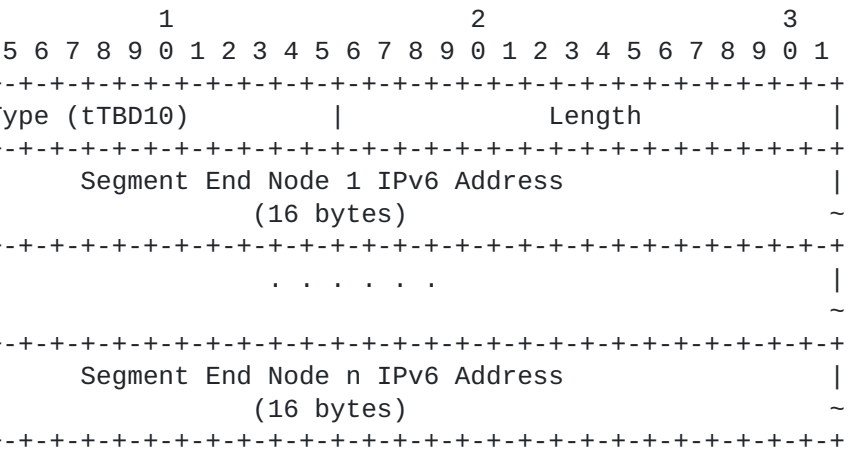
The format of IPv4 SEGMENT-END-NODE-LIST TLV is as follows:



Format of IPv4 SEGMENT-END-NODE-LIST TLV

The IPv4 SEGMENT-END-NODE-LIST contains n segment node IPv4 addresses. An IPv4 SEGMENT-END-NODE-LIST is also called an IPv4 SEGMENT-END-NODES.

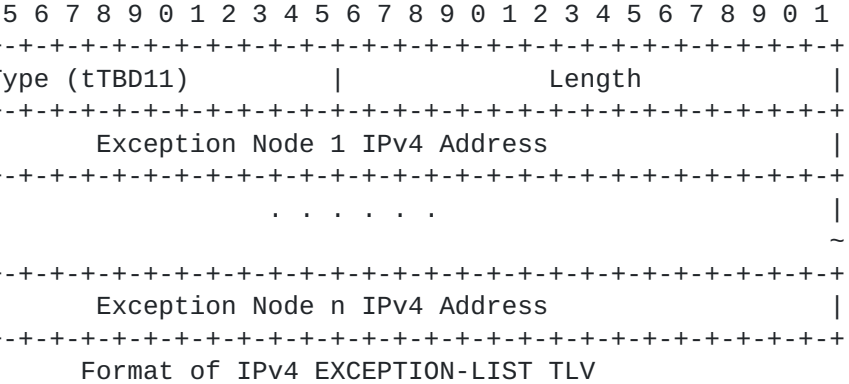
The format of IPv6 SEGMENT-END-NODE-LIST TLV is as follows:



Format of IPv6 SEGMENT-END-NODE-LIST TLV

The IPv6 SEGMENT-END-NODE-LIST contains n segment end node IPv6 addresses. An IPv6 SEGMENT-END-NODE-LIST is also called an IPv6 SEGMENT-END-NODES.

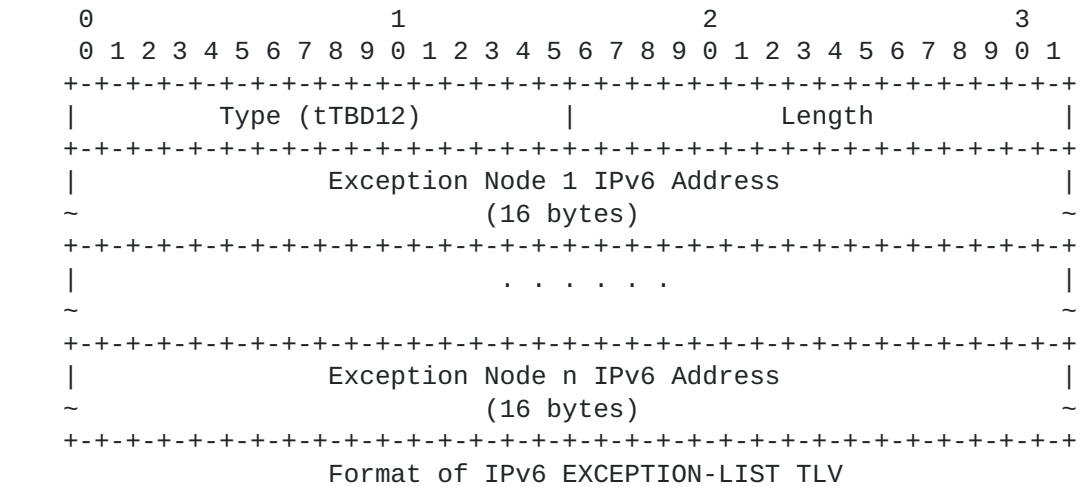
The format of IPv4 EXCEPTION-LIST TLV is as follows:



Format of IPv4 EXCEPTION-LIST TLV

The IPv4 EXCEPTION-LIST contains n node IPv4 addresses in an exception-list. An IPv4 EXCEPTION-LIST is also called an IPv4 EXCEPTION-NODE-LIST.

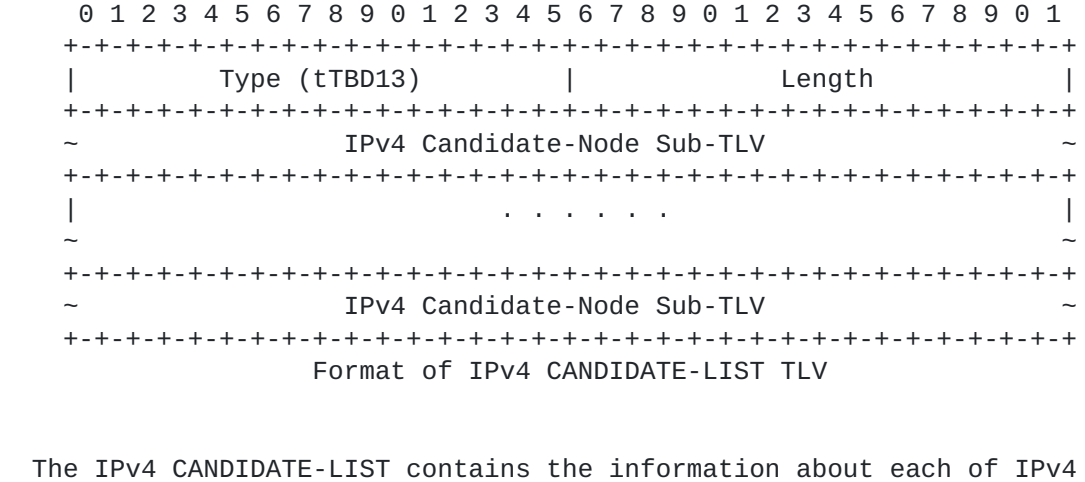
The format of IPv6 EXCEPTION-LIST TLV is as follows:



Format of IPv6 EXCEPTION-LIST TLV

The IPv6 EXCEPTION-LIST contains n node IPv6 addresses in an exception-list. An IPv6 EXCEPTION-LIST is also called an IPv6 EXCEPTION-NODE-LIST.

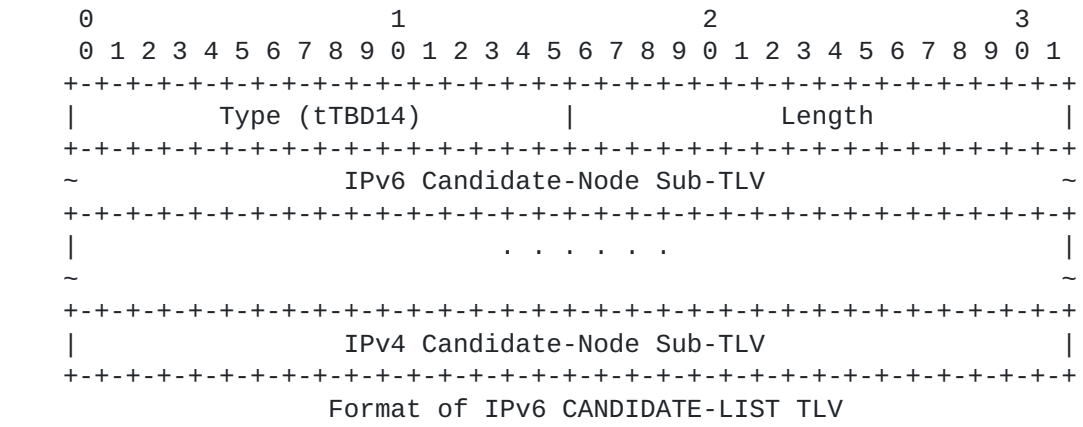
The format of IPv4 CANDIDATE-LIST TLV is as follows:



Format of IPv4 CANDIDATE-LIST TLV

The IPv4 CANDIDATE-LIST contains the information about each of IPv4 nodes in an candidate-list, which is represented in an IPv4 Candidate-Node Sub-TLV.

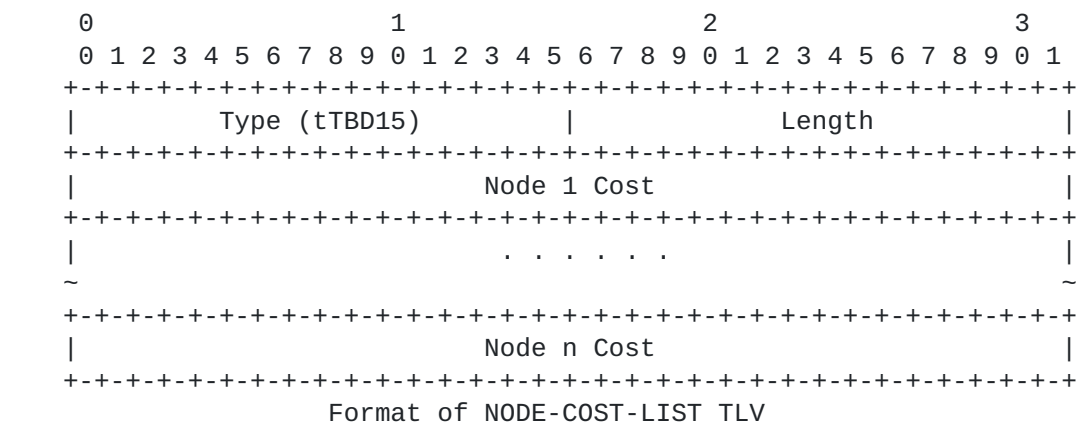
The format of IPv6 CANDIDATE-LIST TLV is as follows:



Format of IPv6 CANDIDATE-LIST TLV

The IPv4 CANDIDATE-LIST contains the information about each of IPv4 nodes in an candidate-list, which is represented in an IPv4 Candidate-Node Sub-TLV.

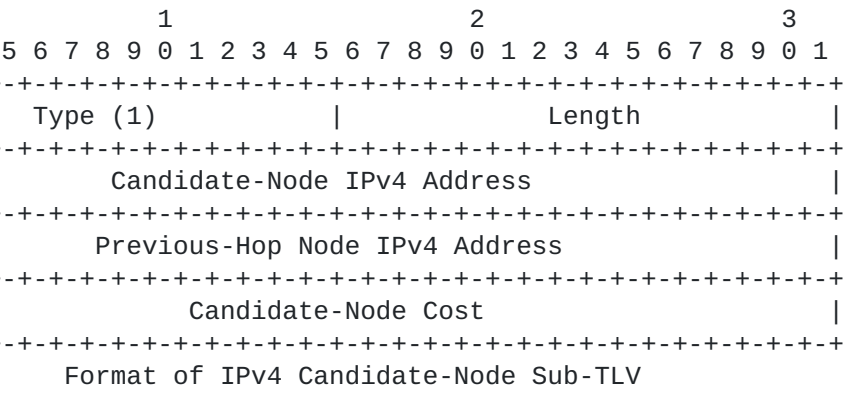
The format of NODE-COST-LIST TLV is as follows:



Format of NODE-COST-LIST TLV

The NODE-COST-LIST contains n costs for n segment end nodes. The type of cost is determined by the value of Optimization Parameter in Controller Request Parameters TLV.

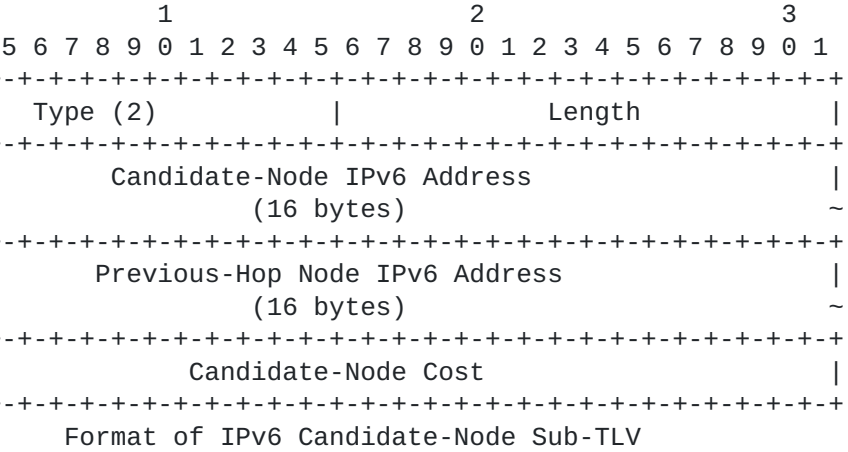
The format of IPv4 Candidate-Node Sub-TLV is as follows:



Format of IPv4 Candidate-Node Sub-TLV

The Candidate-Node IPv4 Address indicates the IPv4 address of the candidate node. The Previous-Hop Node IPv4 Address is the IPv4 address of the previous hop node of the candidate node. The Candidate-Node Cost indicates the cost to the candidate node. The type of the cost is determined by the value of Optimization Parameter in Controller Request Parameters TLV.

The format of IPv6 Candidate-Node Sub-TLV is as follows:

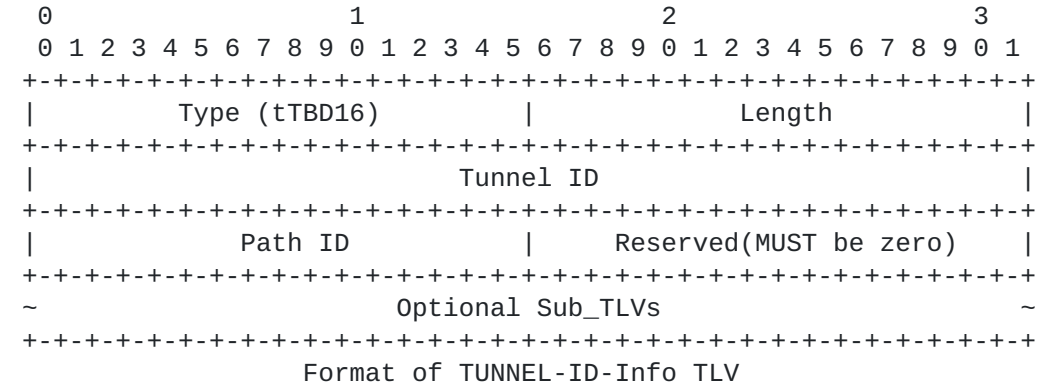


Format of IPv6 Candidate-Node Sub-TLV

The Candidate-Node IPv6 Address indicates the IPv6 address of the candidate node. The Previous-Hop Node IPv6 Address is the IPv6 address of the previous hop node of the candidate node. The Candidate-Node Cost indicates the cost to the candidate node. The type of the cost is determined by the value of Optimization Parameter in Controller Request Parameters TLV.

6.5.6. TUNNEL-ID-Info TLV

The format of TUNNEL-ID-Info TLV is as follows:



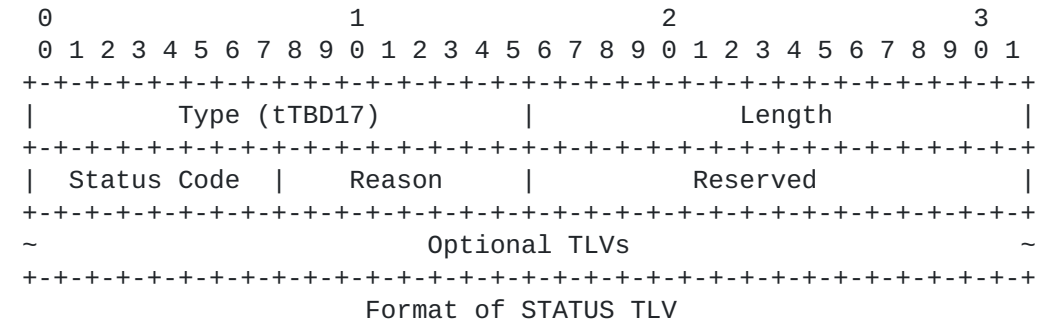
Format of TUNNEL-ID-Info TLV

The Tunnel ID is a 32-bit unique number for identifying a tunnel globally. The Path ID is a 16-bit number for uniquely identifying a path under a tunnel.

When the Path ID is zero, the TUNNEL-ID-Info indicates all the paths or path segments under the Tunnel ID if it is used to indicate paths or path segments.

6.5.7. STATUS TLV

The format of STATUS TLV has following format:



Format of STATUS TLV

The status code (or status for short) in a STATUS may be one of the followings:

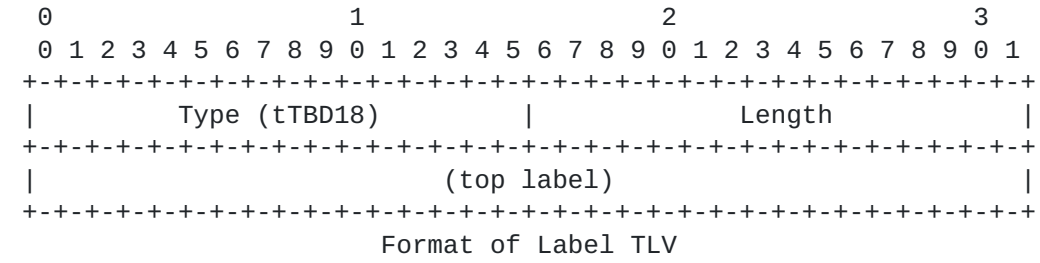
1 (SUCCESS): Indicating a request is successfully finished.

2 (FAIL): Indicating a request can not be finished.

When the status is FAIL, the Reason gives a reason for the failure and the Optional TLVs give some more details about failure.

6.5.8. LABEL TLV

The format of LABEL TLV has following format:



Format of Label TLV

The contents of a LABEL is a single label, encoded in 4 octets.

6.5.9. INTERFACE TLVs

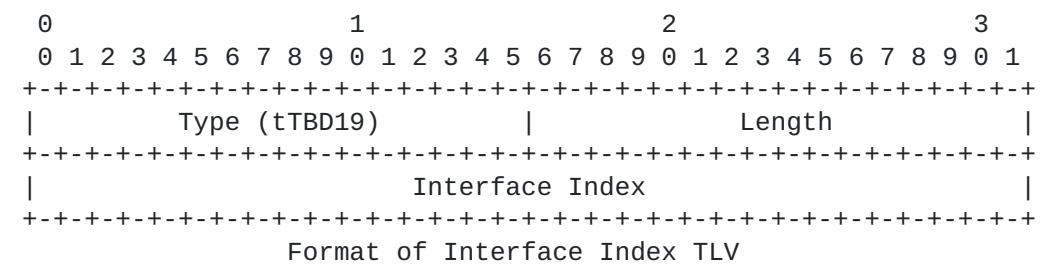
Three TLVs for Interface are defined:

1. Interface Index TLV

2. Interface IPv4 Address TLV

3. Interface IPv6 Address TLV

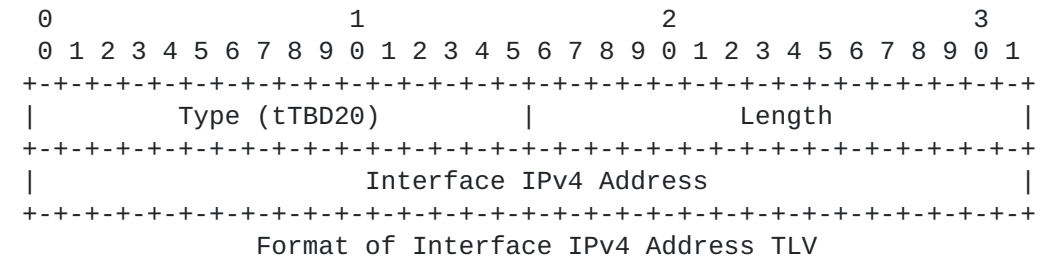
The format of interface index TLV has following format:



Format of Interface Index TLV

The Interface Index is a single interface index, encoded in 4 octets.

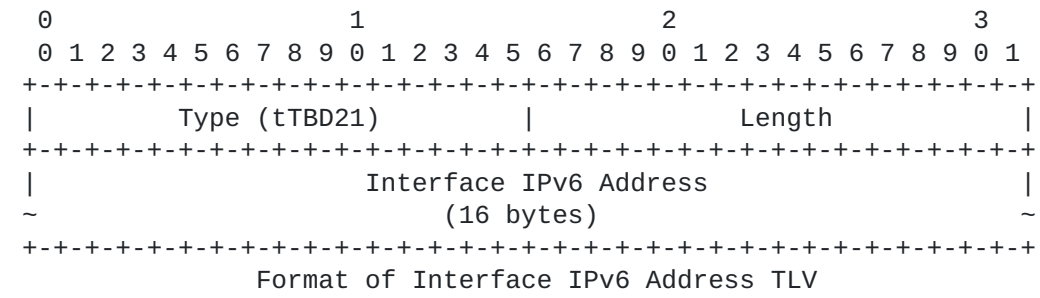
The format of interface IPv4 address TLV has following format:



Format of Interface IPv4 Address TLV

The Interface IPv4 Address is a single interface IPv4 address, encoded in 4 octets.

The format of interface IPv6 address TLV has following format:



Format of Interface IPv6 Address TLV

The Interface IPv6 Address is a single interface IPv6 address, encoded in 16 octets.

6.5.10. CONSTRAINS TLVs

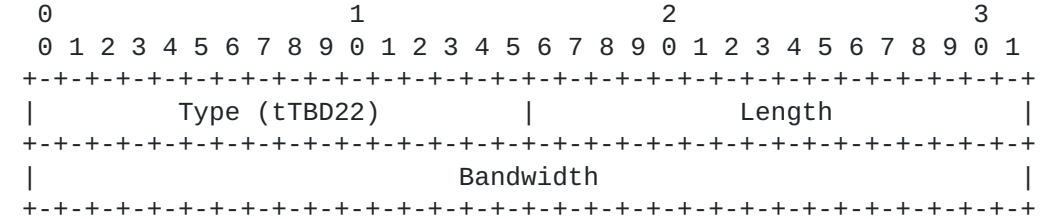
Three TLVs for Constrains are defined:

1. BANDWIDTH TLV

2. LSPA (LSP Attributes) TLV

3. ER (Explicit Route) TLV

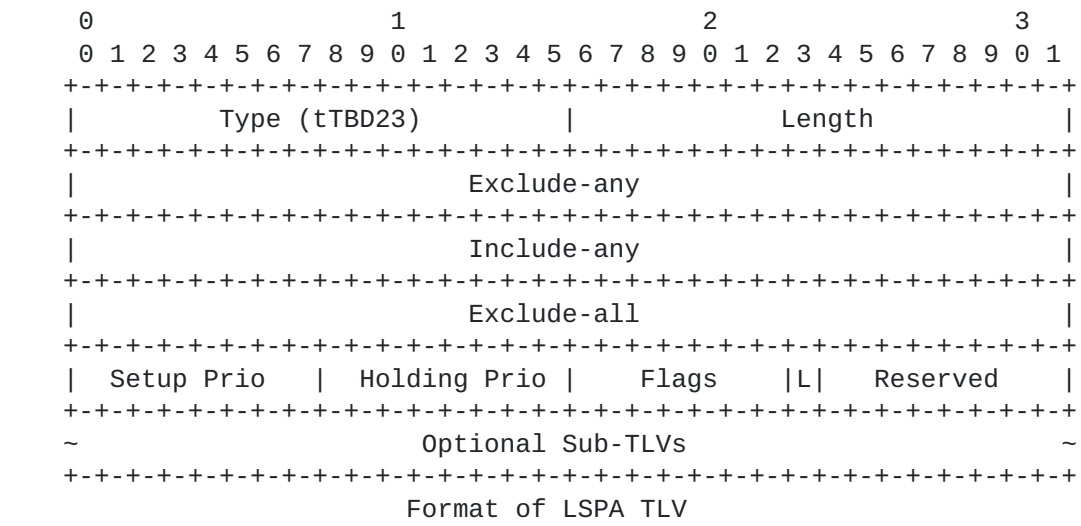
The format of BANDWIDTH TLV has following format:



Format of BANDWIDTH TLV

The Bandwidth is encoded in 32 bits in IEEE floating point format (see [IEEE.754.1985]), expressed in bytes per second. Refer to Section 3.1.2 of [RFC3471] for a table of commonly used values.

The format of LSPA TLV has following format:



Format of LSPA TLV

Setup Prio (Setup Priority - 8 bits): The priority of the TE LSP with respect to taking resources, in the range of 0 to 7. The value 0 is the highest priority. The Setup Priority is used in deciding whether this session can preempt another session.

Holding Prio (Holding Priority - 8 bits): The priority of the TE LSP with respect to holding resources, in the range of 0 to 7. The value 0 is the highest priority. Holding Priority is used in deciding whether this session can be preempted by another session.

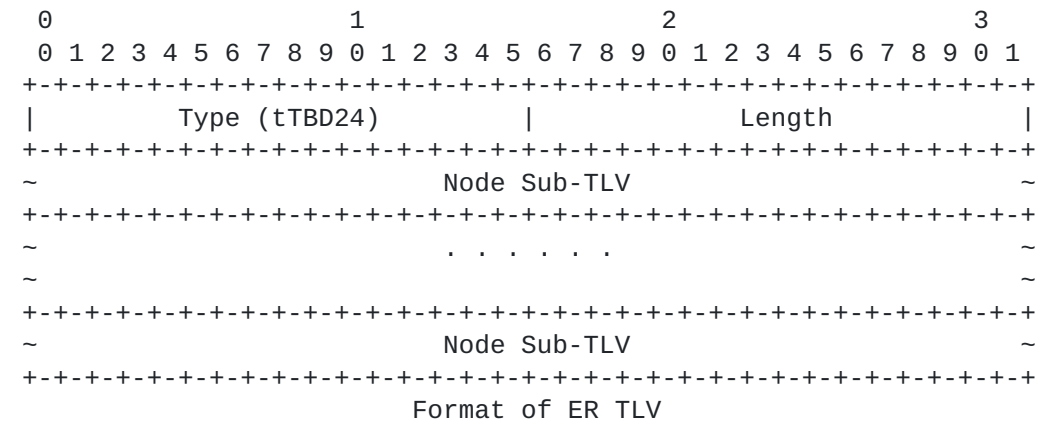
Flags (8 bits)

L flag: Corresponds to the "Local Protection Desired" bit ([RFC3209]) of the SESSION-ATTRIBUTE Object. When set, this means that the computed path must include links protected with Fast Reroute as defined in [RFC4090].

Unassigned flags: MUST be set to zero on transmission and MUST be ignored on receipt.

Reserved (8 bits): This field MUST be set to zero on transmission and MUST be ignored on receipt.

The format of ER TLV is shown below:

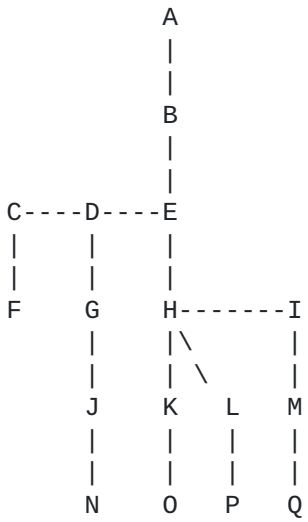


Format of ER TLV

Where the Node Sub-TLVs are defined in the following section.

6.5.11. SPT TLV

The information about a SPT is encoded using explicit route (ER) and secondary explicit route (SER) Sub-TLVs in a compression format. The Figure below illustrates a SPT with A as its root and five leaves F, N, O, P and Q.



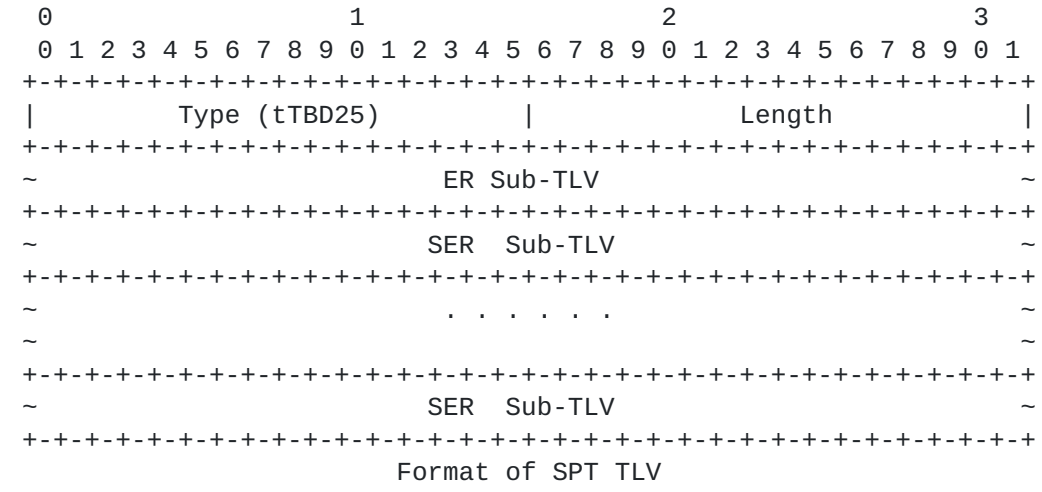
One way to encode the SPT in compression is as follows:

```
   Branch A-F: ER Sub-TLV = {A, B, E, D, C, F, c2F}
   Branch A-N: SER Sub-TLV = {D, G, J, N, c2N}
   Branch A-O: SER Sub-TLV = {E, H, K, O, c2O}
   Branch A-P: SER Sub-TLV = {H, L, P, c2P}
   Branch A-Q: SER Sub-TLV = {H, I, M, Q, c2Q}
```

Where for each of the five leaf nodes, it is followed by the cost to it from the root. For example, leaf node F is followed by cost c2F, which is the cost to F from root A.

In this compressed encoding, the potential repetition of path information for multiple branches that share path segments in the SPT is eliminated.

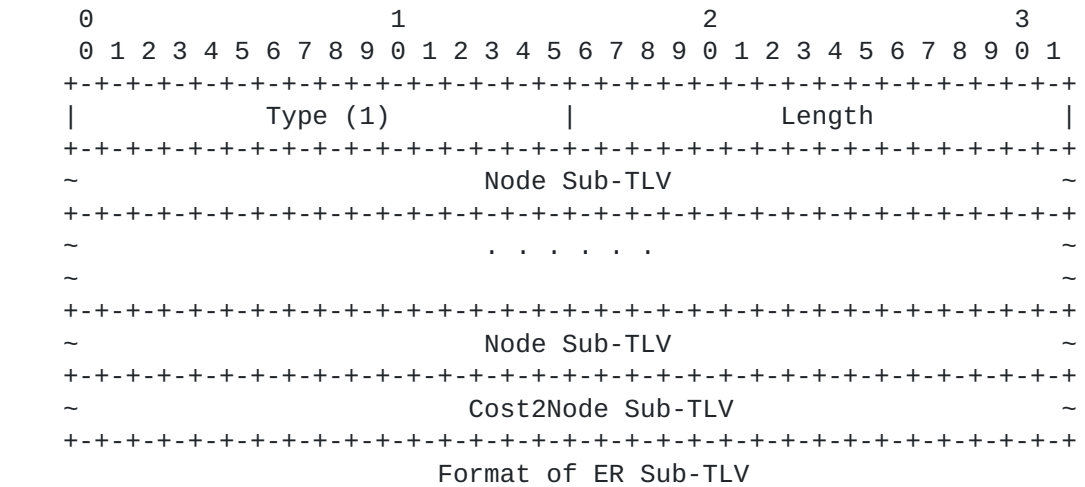
The format of SPT TLV has following format:



Format of SPT TLV

A SPT TLV contains one ER Sub-TLV, which may be followed by a number of SER Sub-TLVs.

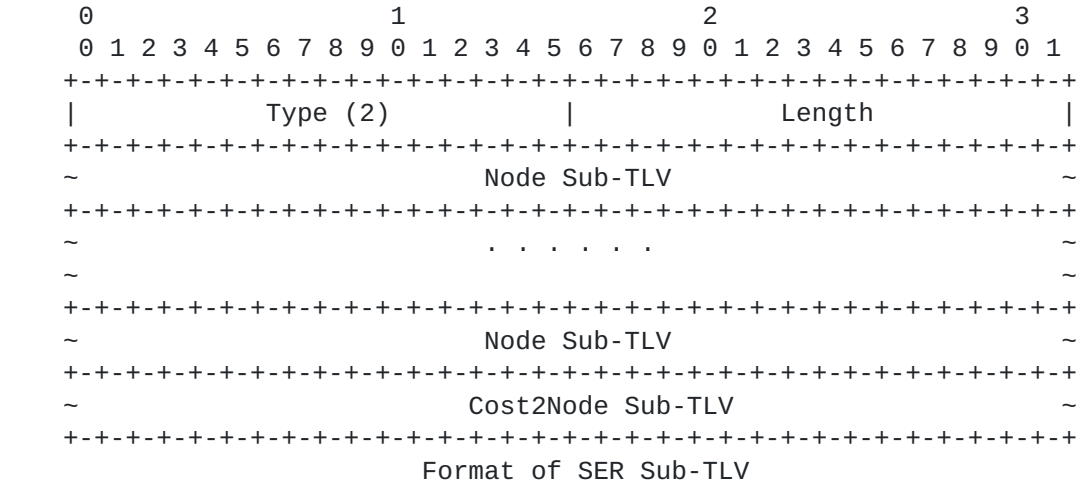
The format of ER Sub-TLV has following format:



Format of ER Sub-TLV

An ER Sub-TLV contains a number of Node Sub-TLVs, each of which represents a node for a branch in a SPT. The branch is from the root of the SPT to a leaf of the SPT. The order of the Node Sub-TLVs in the ER TLV MUST be the same as the order of the nodes in the branch. Following the last Node Sub-TLV is a Cost2Node Sub-TLV, which indicates the cost to the node represented in the last Node Sub-TLV from the root of the SPT.

The format of SER Sub-TLV is the same as that of ER Sub-TLV. It is illustrated below:



Format of SER Sub-TLV

There are three types of Node Sub-TLVs, which are listed as follows:

1 IPv4 prefix
2 IPv6 prefix
3 Autonomous System Number (ASN)

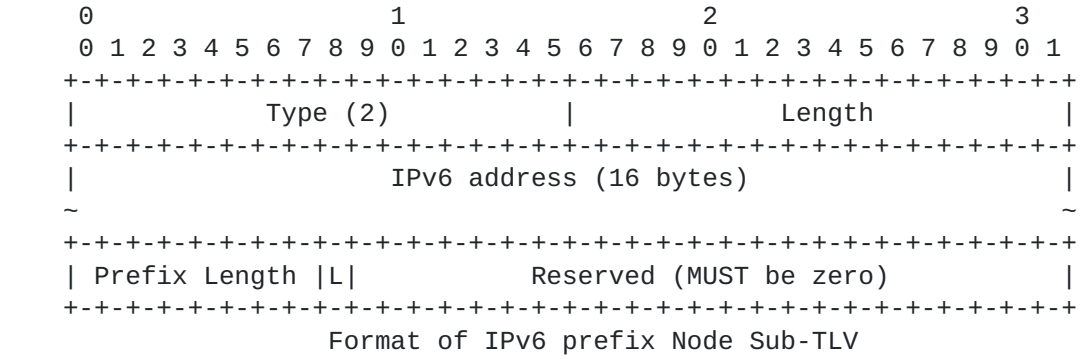
There is one type of Cost2Node Sub-TLV. The type of the cost represented in the Sub-TLV is determined by the Optimization Parameter in Controller Request Parameters TLV.

The format of IPv4 prefix Node Sub-TLV is shown below:

Format of IPv4 prefix Node Sub-TLV

The L bit is set to 1 if the Sub-TLV represents a loose hop. If the bit is not set (i.e., it is 0), the Sub-TLV represents a strict hop.

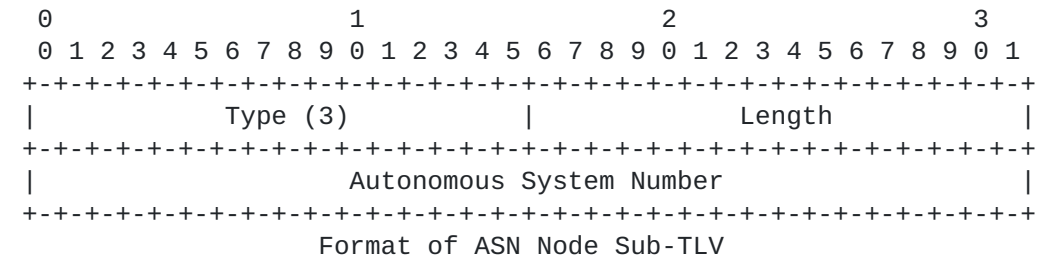
The format of IPv6 prefix Node Sub-TLV is shown below:



Format of IPv6 prefix Node Sub-TLV

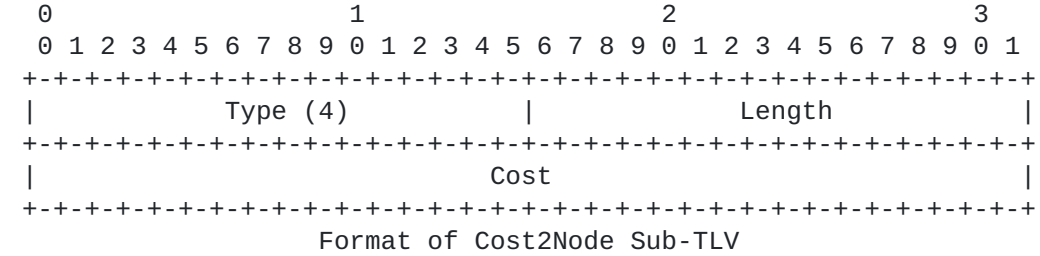
The L bit is set to 1 if the Sub-TLV represents a loose hop. If the bit is not set (i.e., it is 0), the Sub-TLV represents a strict hop.

The format of ASN Node Sub-TLV is shown below:



Format of ASN Node Sub-TLV

The format of Cost2Node Sub-TLV is shown below:



Format of Cost2Node Sub-TLV

7. Security Considerations

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

8. IANA Considerations

This section specifies requests for IANA allocation.

9. Acknowledgement

The authors would like to thank people for their valuable comments on this draft.

10. References

10.1. Normative References

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

[RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, DOI 10.17487/RFC1771, March 1995, <http://www.rfc-editor.org/info/rfc1771>.

[RFC2842] Chandra, R. and J. Scudder, "Capabilities Advertisement with BGP-4", RFC 2842, DOI 10.17487/RFC2842, May 2000, <http://www.rfc-editor.org/info/rfc2842>.

[RFC2858] Bates, T., Rekhter, Y., Chandra, R., and D. Katz, "Multiprotocol Extensions for BGP-4", RFC 2858, DOI 10.17487/RFC2858, June 2000, <http://www.rfc-editor.org/info/rfc2858>.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <http://www.rfc-editor.org/info/rfc3107>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <http://www.rfc-editor.org/info/rfc3209>.

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <http://www.rfc-editor.org/info/rfc3473>.

[RFC3477] Kompella, K. and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, DOI 10.17487/RFC3477, January 2003, <http://www.rfc-editor.org/info/rfc3477>.

[RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, DOI 10.17487/RFC4090, May 2005, <http://www.rfc-editor.org/info/rfc4090>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <http://www.rfc-editor.org/info/rfc4760>.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <http://www.rfc-editor.org/info/rfc5492>.

[RFC5392] Chen, M., Zhang, R., and X. Duan, "OSPF Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5392, DOI 10.17487/RFC5392, January 2009, <http://www.rfc-editor.org/info/rfc5392>.

[RFC5316] Chen, M., Zhang, R., and X. Duan, "ISIS Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5316, DOI 10.17487/RFC5316, December 2008, <http://www.rfc-editor.org/info/rfc5316>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <http://www.rfc-editor.org/info/rfc5305>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <http://www.rfc-editor.org/info/rfc3630>.

10.2. Informative References

[RFC1136] Hares, S. and D. Katz, "Administrative Domains and Routing Domains: A model for routing in the Internet", RFC 1136, DOI 10.17487/RFC1136, December 1989, <http://www.rfc-editor.org/info/rfc1136>.

[RFC6805] King, D., Ed. and A. Farrel, Ed., "The Application of the Path Computation Element Architecture to the Determination of a Sequence of Domains in MPLS and GMPLS", RFC 6805, DOI 10.17487/RFC6805, November 2012, <http://www.rfc-editor.org/info/rfc6805>.

Authors' Addresses

Huaimo Chen
Huawei Technologies
Boston, MA,
USA

EMail: Huaimo.chen@huawei.com

Susan Hares
Huawei Technologies
Karnataka

EMail: shares@ndzh.com

Yi Yang
Sockrate
USA

EMail: yyang1998@gmail.com

Yanhe Fan
Casa Systems, Inc.
USA

EMail: yfan@casa-systems.com

Mehmet Toy
Verizon
USA

EMail: mehmet.toy@verizon.com

Zhenqiang Li
China Mobile
No.32 Xuanwumenxi Ave., Xicheng District
Beijing 100032
P.R. China

EMail: li_zhenqiang@hotmail.com

Lei Liu
Fujitsu
USA

EMail: lliu@us.fujitsu.com