MPLS Working Group R. Torvi
Internet-Draft R. Bonica
Intended status: Informational Juniper Networks
Expires: April 26, 2015 M. Conn
D. Pacella
L. Tomotaki
M. Wygant
Verizon
October 23, 2014

LSP Self-Ping
draft-bonica-mpls-self-ping-02

Abstract

This memo describes LSP Self-ping. Ingress LSR’s can use LSP Self-ping to verify that an LSP is ready to carry traffic.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

An ingress Label Switching Router (LSR) can use RSVP-TE [RFC3209] to establish an MPLS Label Switched Path [RFC3032]. The following paragraphs provide an overview of RSVP-TE procedures.

The ingress LSR calculates an explicit path between itself and an egress LSR. It then formats an RSVP PATH message, including an Explicit Route Object (ERO). The ERO represents the explicit path between the ingress and egress LSRs.

The ingress LSR forwards the PATH message in the direction of the egress LSR, following the path defined by the ERO. Each transit LSR that receives the PATH message executes admission control procedures. If the transit LSR admits the LSP, it reserves bandwidth (if necessary) and sends the PATH message downstream, to the next node in the ERO.

When the egress LSR receives the PATH message, it binds a label to the LSP. The label can be implicit null, explicit null, or non-null. The egress LSR then installs forwarding state (if necessary), and constructs an RSVP RESV message. The RESV message includes a Label Object containing the label that has been bound to the LSP.

The egress LSR sends the RESV message upstream towards the ingress LSR. The RESV message visits the same transit LSRs that the PATH message visited, but in reverse order. Each transit LSR binds a label to the LSP, updates its forwarding state and updates the RESV message. As a result, the RESV message contains a Label Object and the Label Object contains the label that has been bound to the LSP. Next, the transit LSR sends the RESV message upstream, along the explicit path.

The ingress LSR receives the RESV message and installs forwarding state. Once the ingress LSR installs forwarding state it can forward traffic through the LSP.

An implementation can optimize the procedure described above by allowing LSRs to send a RESV messages upstream before installing forwarding state. This optimization is desirable, because it allows LSRs to install forwarding state in parallel, thus accelerating the process of LSP signaling and setup. However, this optimization creates a race condition. When the ingress LSR receives a RESV message, some downstream LSRs may have not yet completed the process of forwarding state installation. If the ingress sends traffic over the LSP, the traffic will be black-holed until forwarding state has been installed on all downstream LSRs.

The ingress LSP can prevent back-holing by verifying the LSPs readiness to carry traffic before forwarding traffic through it. Ingress LSRs can use LSP Self-Ping to verify that an LSP is ready to carry traffic.

LSP Self-ping is an extremely lightweight mechanism, designed to perform well when control plane resources are scarce. Therefore, LSP Self-ping consumes no control plane resources on transit or egress LSRs.

This memo describes LSP Self-ping.

2. LSP Self Ping Procedures

In order to verify that an LSP is ready to carry traffic, the ingress LSR creates a short-lived LSP Self-ping session. All session state is maintained locally on the ingress LSR. Session state includes the following:

- Session-id: A 32-bit number that identifies the session

- verification-status: A boolean variable indicating whether LSP readiness has been verified. The initial value of this variable is FALSE.

- retries: The number of times that the ingress LSR probes the LSP before giving up. The initial value of this variable is determined by configuration.

- retry-timer: The number of milliseconds that the LSR waits after probing the LSP. The initial value of this variable is determined by configuration.

The ingress LSR executes the following procedure until verification-status equals TRUE or retries is less than 1:

- Format a MPLS Echo [RFC4379] message
- Send the MPLS Echo message through the LSP under test
- Set a timer to expire in retry-timer milliseconds
- Wait until either a) a MPLS Echo message associated with the session returns or b) the timer expires. If an MPLS Echo message associated with the session returns, set verification-status to TRUE. Otherwise, decrement retries. Optionally, increase the value of retry-timer according to an appropriate back off algorithm.

As per [RFC4379], the MPLS Echo message is encapsulate in a User Datagram Protocol (UDP) [RFC0768] header. If the protocol messages used to establish the LSP were delivered over IPv4 [RFC0791], the UDP datagram is encapsulated in an IPv4 header. If the protocol messages used to establish the LSP were delivered over IPv6 [RFC2460], the UDP datagram is encapsulated in an IPv6 header.

In either case, message contents are as follows:

- IP Source Address is configurable. By default, it is the address of the egress LSR
- IP Destination Address is the address of the ingress LSR
- IP Time to Live (TTL) / Hop Count is 255
- IP DSCP is configurable. By default, it is equal to CS6 (Ox48) [RFC4594]
- UDP Source Port is 3503
- UDP Destination Port is 3503
- MPLS Echo Global Flags are clear (i.e., set to 0)
- MPLS Echo Type is equal to "MPLS Echo Reply" (2)
- MPLS Echo Reply Mode is "Reply via an IPv4/IPv6 UDP packet" (2)
- MPLS Echo Senders Handle is equal to the Session-ID
- MPLS Echo Sequnce Number is equal to retries

The reader should note that the ingress LSR probes the LSP by sending an MPLS Echo message, addressed to itself, through the LSP. The egress LSR forwards the MPLS Echo message back to the ingress LSR, exactly as it would forward any other packet.

If the LSP under test is ready to carry traffic, the egress LSR receives the MPLS Echo message. The MPLS Echo message can arrive at the egress LSR with or without an MPLS header, depending on whether the LSP under test executes penultimate hop-popping procedures. If the MPLS Echo message arrives at the egress LSR with an MPLS header, the egress LSR removes that header.

The egress LSR forwards the MPLS Echo message to its destination, the ingress LSR. The egress LSR forwards the MPLS Echo message exactly as it would forward any other packet. If the egress LSR’s most preferred route to the ingress LSR is through an LSP, the egress LSR forwards the MPLS Echo message through that LSP. However, if the egress LSR’s most preferred route to the ingress LSR is not through an LSP, the egress LSR forwards the MPLS Echo message without MPLS encapsulation.

If the ingress LSR receives an MPLS Echo message with Senders Handle equal to the Session-ID, it sets the verification-status to TRUE. The Sequence Number does not have to match the last Sequence Number sent.

When an LSP Self-ping session terminates, it returns the value of verification-status to the invoking protocol. For example, assume that RSVP-TE invokes LSP Self-ping as part of the LSP set-up procedure. If LSP Self-ping returns TRUE, RSVP-TE makes the LSP under test available for forwarding. However, if LSP Self-ping returns FALSE, RSVP-TE takes appropriate remedial actions.

LSP Self-ping fails if all of the following conditions are true:

o The Source Address of the MPLS Echo message is equal to its default value (that is, the address of the egress LSR)

o The penultimate hop pops the MPLS label

o The egress LSR executes Unicast Reverse Path Forwarding (uRPF) procedures

In this scenario and in similar scenarios, the egress LSR discards the MPLS Echo message rather than forwarding it. In such scenarios, the calling application can set the source address to a more appropriate value.

3. Rejected Approaches

In a rejected approach, the ingress LSR uses LSP-Ping, exactly as described in [RFC4379] to verify LSP readiness to carry traffic. This approach was rejected for the following reasons.

While an ingress LSR can control its control plane overhead due to LSP Ping, an egress LSR has no such control. This is because each ingress LSR can, on its own, control the rate of the LSP Ping originated by the LSR, while an egress LSR must respond to all the LSP Pings originated by various ingresses. Furthermore, when an MPLS Echo Request reaches an egress LSR it is sent to the control plane of the egress LSR, which makes egress LSR processing overhead of LSP Ping well above the overhead of its data plane (MPLS/IP forwarding). These factors make LSP Ping problematic as a tool for detecting LSP readiness to carry traffic when dealing with a large number of LSPs.

By contrast, LSP Self-ping does not consume any control plane resources at the egress LSR, and relies solely on the data plane of the egress LSR, making it more suitable as a tool for checking LSP readiness when dealing with a large number of LSPs.

In another rejected approach, the ingress LSR does not verify LSP readiness. Alternatively, it sets a timer when it receives an RSVP RESV message and does not forward traffic through the LSP until the timer expires. This approach was rejected because it is impossible to determine the optimal setting for this timer. If the timer value is set too low, it does not prevent black-holing. If the timer value is set too high, it slows down the process of LSP signalling and setup.

Moreover, the above-mentioned timer is configured on a per-router basis. However, its optimum value is determined by a network-wide behavior. Therefore, changes in the network could require changes to the value of the timer, making the optimal setting of this timer a moving target.

4. IANA Considerations

This document makes no request of IANA.

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

5. Security Considerations

MPLS Echo messages are easily forged. Therefore, an attacker can send the ingress LSR a forged MPLS Echo message, causing the ingress LSR to terminate the LSP Self-ping session prematurely.

6. Acknowledgements

Thanks to Yakov Rekhter, Ravi Singh, Eric Rosen, Eric Osborne and Nobo Akiya for their contributions to this document.

7. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

Authors’ Addresses

Ravi Torvi
Juniper Networks

Email: rtorvi@juniper.net

Ron Bonica
Juniper Networks

Email: rbonica@juniper.net

Michael Conn
Verizon

Email: michael.e.conn@verizon.com

Dante Pacella
Verizon

Email: dante.j.pacella@verizon.com

Luis Tomotaki
Verizon

Email: luis.tomotaki@verizon.com

Mark Wygant
Verizon

Email: mark.wygant@verizon.com

MPLS
Internet-Draft
Intended status: Informational
Expires: April 27, 2015

S. Bryant
C. Pignataro
Cisco Systems

October 24, 2014

MPLS Flow Identification
draft-bryant-mpls-flow-ident-00

Abstract

This memo discusses the desired capabilities for MPLS flow identification. The key application that needs this is in-band performance monitoring of user data packets.

Status of This Memo

This Internet-Draft will expire on April 27, 2015.

Copyright Notice

Table of Contents

1. Introduction

This memo discusses the desired capabilities for MPLS flow identification. The key application that needs this is in-band performance monitoring of user data packets.

There is a need to identify flows in MPLS networks for applications such as packet loss and packet delay measurement. A method of loss and delay measurement in MPLS networks was defined in [RFC6374]. However this work needs to be extended to deal with different granularities of flow and to address a number of the multi-point cases in which a number of ingress LSRs could send to one or more destinations.

Improvements in link and transmission technologies mean that it is difficult to assess a loss using synthetic traffic due to the very low loss rate in normal operation. That together with more demanding service level requirements mean that network operators need to be able to measure the loss of the actual user data traffic. Any technique deployed needs to be transparent to the end user, and it needs to be assumed that they will not take any active part in the measurement process. Indeed it is important that any flow identification technique be invisible to them and that no remnant of the identification of measurement process leak into their network.

2. Loss Measurement Considerations

Modern networks normally drop very few packets, thus packet loss measurement are highly sensitive to counter errors. Without some form of coloring or batch marking such as that proposed in [I-D.tempia-opsawg-p3m] it may not be possible to achieve the required accuracy in the loss measurement of customer data traffic. Where accuracy better than the data link loss performance of a modern optical network is required it may be economically advantage to include temporal marking.

Where this level of accuracy is required and the traffic between a source-destination pair is subject to ECMP a demarcation mechanism is needed to group the packets into batches. The packet accounting mechanism is then able to operate on a batch of packets which can be accounted for at both the packet ingress and the packet egress. Errors in the accounting are particularly acute in LSPs subjected to ECMP because the network transit time will be different for the various ECMP paths since:

a. The packets may traverse different sets of LSRs.

b. The packets may depart from different interfaces on different line cards on LSRs

c. The packets may arrive at different interfaces on different line cards on LSRs.

A consideration in modifying the identity label to indicate the batch is the impact that this has on the path chosen by the ECMP mechanism. When the member of the ECMP path set is chosen by deep packet inspection a change of colour represented by a change of identity label will have no impact on the ECMP path. Where the path member is chosen by reference to an entropy label [RFC6790] then provided that the entropy label is higher in the stack than the label that is changing colour again there will be no change to the chosen ECMP path. ECMP is so pervasive in multi-point to (multi-) point networks that some method of avoiding accounting errors introduced by ECMP needs to be supported.

3. Units of identification

The most basic unit of identification is the identity of the node processed the packet on its entry to the MPLS network. However, the required unit of identification may vary depending on the use case for accounting, performance measurement or other types of packet observations. In particular note that there mat be a need to impose identify at several different layers of the MPLS label stack.

This document considers follow unit of identifications:

- Per source LSR - everything from one source is aggregated.

- Per group of LSPs chosen by an ingress LSR - an ingress LSP aggregates group of LSPs (ex: all LSPs of a tunnel).

- Per LSP - the basic form.

- Per flow [RFC6790] within an LSP - fine graining method.

Note that a finer grained identity resolution is needed when there is a need to perform these operations on a flow not readily identified by some other element in the label stack. Such fine grained resolution may be possible by deep packet inspection, but this may not always be possible, or it may be desired to minimise processing costs by doing only in entry to the network, and adding a suitable identifier to the packet for reference by other network elements. An example of such a fine grained case might be traffic from a specific application, or from a specific application from a specific source, particularly if matters related to service level agreement or application performance were being investigated.

We can thus characterize the identification requirement in the following broad terms:

- There needs to be some way for an egress LSR to identify the ingress LSR with an appropriate degree of scope. This concept is discussed further in Section 5.

- There needs to be a way to identify a specific LSP at the egress node. This allows for the case of instrumenting multiple LSPs operate between the same pair of nodes. In such cases the identity of the ingress LSR is insufficient.

- In order to conserve resources such as labels, counters and/or compute cycles it may be desirable to identify an LSP group so that a operation can be performed on the group as an aggregate.

- There needs to be a way to identify a flow within an LSP. This is necessary when investigating a specific flow that has been aggregated into an LSP.

The method of determining which packets constitute a flow is outside the scope of this memo.

4. Types of LSP

We need to consider a number of types of LSP. The two simplest types to monitor are point to point LSPs and point to multi-point LSPs. The ingress LSR for a point to point LSP, such as those created using the RSVP-TE signalling protocol, or those that conform to the MPLS-TP may be identified by inspection of the top label in the stack, since at any PE or P router on the path this is unique to the ingress-egress pair at every hop at a given layer in the LSP hierarchy. Provided that penultimate hop popping is disabled, the identity of the ingress LSR of a point to point LSP is available at the egress LSR and thus determining the identity of the ingress LSR must be regarded as a solved problem. Note however that the identity of a flow cannot to be determined without further information.

In the case of a point to multi-point LSP the identity of the ingress LSR may also be inferred from the top label. However it is not possible to identify a flow from the top label, nor is it possible to directly identify the ingress LSR since there may be many point to multi-point LSP originating at that LSR. In designing any solution it is desirable that a common flow identity solution be used for both point to point and point to multi-point LSP types. Similarly it is desirable that a common method of LSP group identification be used.

In the above cases, an explicit non-null label is needed to provide context at the egress LSR. This is widely supported MPLS feature.

A more interesting case, and the core purpose of this memo, is the case of a multi-point to point LSP. In this case the same label is normally used by multiple ingress or upstream LSRs and hence source identification is not possible by inspection of the top label by egress LSRs. It is therefore necessary for a packet to be able to explicitly convey any of the identity types described in Section 3.

Similarly, in the case of a multi-point to multi-point LSP the same label is normally used by multiple ingress or upstream LSRs and hence source identification is not possible by inspection of the top label by egress LSRs. The various types of identity described in Section 3 are again needed. Note however, that the scope of the identity may be constrained to be unique within the set of multi-point to multi-point LSPs terminating on any common node.

Any method of identity must not consume an excessive number of unique labels, nor result in an excessive increase in the size of the label stack (Section 7).

5. Network Scope

The scope of identification can be constrained to the set of flows that are uniquely identifiable at an ingress LSR, or some aggregation thereof. There is no question of an ingress LSR seeking assistance from outside the MPLS domain.

In any solution that constrains itself to carrying the required identity in the MPLS label stack rather than in some different associated data structure, constraints on the label stack size imply that the scope of identity reside within that MPLS domain. For similar reasons the identity scope of a component of an LSP should be constrained to the scope of that LSP.

6. Backwards Compatibility

In any network it is unlikely that all LSRs will have the same capability to support the methods of identification discussed in this memo. It is therefore an important constraint on any identity solution that it is backwards compatible with deployed MPLS equipment to the extent that deploying the new feature will not disable anything that currently works on a legacy equipment.

This is particularly the case when the deployment is incremental or when the feature is not required for all LSRs or all LSPs. Thus in broad the flow identification design must support the co-existence of LSRs that can and cannot identify the traffic components described in . (Section 3). In addition the identification of the traffic components described in Section 3 needs to be an optional feature that is disabled by default. As a design simplification, a solution may require that all egress LSRs of a point to multipoint or a multipoint to multipoint LSP to support the identification type in use so that a single packet can be correctly processed by all egress devices. The corollary of this last point is that either all egress LSRs are enabled to support the required identity type, or none of them are.

7. Dataplane

There is a huge installed base of MPLS equipment, typically this type of equipment remains in service for an extended period of time, and in many cases hardware constraints mean that it is not possible to upgrade its dataplane functionality. Changes to the MPLS data plane are therefore expensive to implement, add complexity to the network, and may significantly impact the deployability of a solution that requires such changes. For these reasons, the MPLS designers have set a very high bar to changes to the MPLS data plane, and only a very small number have been adopted. Hence, it is important that the method of identification must minimize changes to the MPLS data plane. Ideally method(s) of identification that require no changes to the MPLS data plane should be given preferential consideration. If a method of identification makes a change to the data plane is chosen it will need to have a significant advantage over any method that makes no change, and the advantage of the approach will need to be carefully evaluated and documented. If a change is necessary to the MPLS data plane proves necessary, it should be (a) be as small a change as possible and (b) be a general purpose method so as to maximise its use for future applications. It is imperative that, as far as can be foreseen, any necessary change made to the MPLS data plane does not impose any foreseeable future limitation on the MPLS data plane.

Stack size is an issue with many MPLS implementations both as a result of hardware limitations, and due to the impact on networks and applications where a large number of small payloads need to be transported In particular one MPLS payload may be carried inside another. For example one LSP may be carried over another LSP, or a PW or similar multiplexing construct may be carried over an LSP and identification may be required at both layers. Of particular concern is the implementation of low cost edge LSRs that for cost reasons have a significant limit on the number of LSEs that they can impose or dispose.

The MPLS data plane design provides only a tiny number of reserved labels, it is therefore core to the MPLS design philosophy that this scarce resource is only used when it is absolutely necessary. Using a single LSE reserved or special purpose label to encode flow identity thus requires two stack entries. A larger special purpose labels space is available [RFC7274] but this requires two labels stack entries for the reserved label itself and hence a total of three label stack entries to encode the flow identity.

The use of special purpose labels (SPL) [RFC7274]as part of a method to encode the identity information therefore has a number of undesirable implications for the data plane and hence whilst a solution may use SPL(s), methods that do not require SPLs need to be carefully considered.

8. Control Plane

Any flow identity design should both seek to minimise the complexity of the control plane and should minimise the amount of label co-ordination needed amongst LSRs.

9. Manageability Considerations

This will be provided in a future version of this document.

10. Privacy Considerations

The inclusion of originating and/or flow information in a packet provides more identity information and hence potentially degrades the privacy of the communication. Recent IETF concerns on pervasive monitoring would lead it to prefer a solution that does not degrade the privacy of user traffic below that of an MPLS network not implementing the flow identification feature. The minimizing the scope of the identity indication can be useful in minimizing the observability of the flow characteristics.

11. Security Considerations

Any solution to the flow identification needs must not degrade the security of the MPLS network below that of an equivalent network not deploying the specified identity solution. Propagation of identification information outside the MPLS network imposing it must be disabled by default. Any solution should provide for the restriction of the identity information to those components of the network that need to know it. It is thus desirable to limit the knowledge of the identify of an endpoint to only those LSRs that need to participate in traffic flow.

12. IANA Considerations

EDITOR’S NOTE: This section may be removed on publication

This memo has no IANA considerations.

13. Acknowledgements

The authors thank Nobo Akiya (nobo@cisco.com), Nagendra Kumar Nainar (naikumar@cisco.com) and George Swallow (swallow@cisco.com) for their comments.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

14.2. Informative References

[I-D.tempia-opsawg-p3m]
Capello, A., Cociglio, M., Castaldelli, L., and A. Bonda, "A packet based method for passive performance monitoring", draft-tempia-opsawg-p3m-04 (work in progress), February 2014.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, September 2011.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, June 2014.

Authors’ Addresses

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

Carlos Pignataro
Cisco Systems

Email: cpignata@cisco.com

Network Working Group
Internet Draft
Intended status: Standards Track

Expires: April 27, 2015

Chandra Ramachandran (Ed)
Yakov Rekhter (Ed)
Juniper Networks

October 27, 2014

Refresh Interval Independent FRR Facility Protection
draft-chandra-mpls-enhanced-frr-bypass-00

Status of this Memo

Copyright Notice

Abstract

This document defines RSVP-TE extensions to facilitate refresh-interval independent FRR facility protection.

Conventions used in this document

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

Table of Contents

1. Introduction

The facility backup protection mechanism is one of two methods discussed in [RFC4090] for enabling the fast reroute of traffic onto backup LSP tunnels in 10s of milliseconds, in the event of a failure. This document discusses a few shortcomings with some of the refresh-interval reliant procedures proposed for this method in [RFC4090]. These shortcomings come to the fore under scaled conditions and get highlighted even further when large RSVP refresh intervals are used. The RSVP-TE extensions defined in this document will enhance the facility backup protection mechanism by making the corresponding procedures refresh-interval independent.

2. Motivation

The primary bottleneck that needs to be overcome in order to scale RSVP-TE implementation to establish and maintain in the order of multiple 100K Label Switched Paths (LSPs) is the rate of RSVP protocol messages that would be required to handle the scale of LSPs. RSVP protocol message rate is influenced by both triggered and periodic messages. The facility protection mechanism is the FRR method of choice in scaled scenarios. The timely establishment of backup LSP after failure is critical to keep the LSP state refreshed on routers downstream of the failure. It should be noted that while timely establishment of backup LSPs after failure is a problem on its own, the requirement of RSVP protocol to periodically refresh existing LSP states exacerbates the problem.

One common and straightforward mechanism to mitigate the RSVP message rate problem is to increase the refresh interval of LSP states so that the routers may prioritize backup LSP establishment and other triggered messages. If large refresh time can be complemented with RSVP refresh reduction extensions defined in [RFC2961], then RSVP-TE implementations can use these extensions to avoid rapid retransmits to reliably convey any new state or state change to neighboring router and avoid re-sending the entire message during refresh to neighboring router. Even though the combination of large refresh time and reliable message delivery could be a potential solution, there are some shortcomings if this combination is applied to facility protection specified in [RFC4090].

3. Problem Description

In the topology illustrated in Figure 1, consider a large number of LSPs from A to D transiting B and C. Assume that refresh interval has been configured to be large of the order of minutes and refresh reduction extensions are enabled on all routers.

```
                                [E]
                               /   \
                              /     \
                             /       \
                            /         \
                           /           \
                          /             \
                        [A]-----[B]-----[C]-----[D]
                                  \             /
                                   \           /
                                    \         /
                                     \       /
                                      \     /
                                       \   /
                                        [F]
```

Figure 1: Example Topology

Also assume that node protection has been configured for the LSPs and the LSPs are protected by each router in the following way

- A has made node protection available using bypass LSP A -> E -> C; A is the Point of Local Repair (PLR) and C is Node Protecting Merge Point (NP-MP)

- B has made node protection available using bypass LSP B -> F -> D; B is the PLR and D is the NP-MP

- C has made link protection available using bypass LSP C -> B -> F -> D; C is the PLR and D is the LP-MP

In the above condition, assume that B-C link fails. The following is the sequence of events that is expected to occur for all protected LSPs under normal conditions.

1. B performs local repair and re-directs LSP traffic over the bypass LSP B -> F -> D.
2. B also creates backup state for the LSP and triggers sending of backup LSP state to D over the bypass LSP B -> F -> D.
3. D receives backup LSP states and merges the backups with the protected LSPs.
4. As the link on C over which the LSP states are refreshed has failed, C will no longer receive state refreshes. Consequently the protected LSP states on C will time out and C will send tear down message for all LSPs.

While the above sequence of events has been described in [RFC4090], there are a few problems for which no mechanism has been specified explicitly.

- If the protected LSP on C times out before D receives signaling for the backup LSP, then D would receive PathTear from C prior to receiving signaling for the backup LSP, thus resulting in deleting the LSP state. This would be possible at scale even with default refresh time.

- If upon the link failure C is to keep state until its timeout, then with long refresh interval this may result in a large amount of stale state on C. Alternatively, if upon the link failure C is to delete the state and send PathTear to D, this would result in deleting the state on D, thus deleting the LSP. D needs a reliable mechanism to determine whether it is MP or not to overcome this problem.

- If head-end A attempts to tear down LSP after step 1 but before step 2 of the above sequence, then B may receive the tear down message before step 2 and delete the LSP state from its state database. If B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.

- If B fails to perform local repair in step 1, then B will delete the LSP state from its state database without informing D. As B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.

The purpose of this document is to provide solutions to the above problems which will then make it practical to scale up to a large number of protected LSPs in the network.

4. Solution Aspects

The solution consists of five parts.

- Enhance facility protection method defined in [RFC4090] by introducing MP determination mechanism that enables PLR to signal availability of link or node protection to the MP. See section 4.1 for more details.

- Handle upstream link or node failures by cleaning up LSP states if the node has not found itself as MP through MP determination mechanism. See section 4.2 for more details.

- Introduce extensions to enable a router to send tear down message to downstream router that enables the receiving router to conditionally delete its local state. See section 4.3 for more details.

- Enhance facility protection by allowing a PLR to directly send tear down message to MP without requiring the PLR to either have a working bypass LSP or have already refreshed backup LSP state. See section 4.4 for more details.

- Introduce extensions to enable the above procedures to be backward compatible with routers along the LSP path running implementation that do not support these procedures. See section 4.5 for more details.

4.1. Signaling Protection availability for MP determination

4.1.1. PLR Behavior

When protected LSP comes up and if "local protection desired" is set in SESSION_ATTRIBUTE object, each node along the LSP path attempts to make local protection available for the LSP.

- If "node protection desired" flag is set, then the node tries to become a PLR by attempting to create NP-bypass LSP to NNhop node avoiding the Nhop node on protected LSP path. In case node protection could not be made available after some time out, the node attempts to create a LP-bypass LSP to Nhop node avoiding only the link that protected LSP takes to reach Nhop

- If "node protection desired" flag is not set, then the PLR attempts to create a LP-bypass LSP to Nhop node avoiding the link that protected LSP takes to reach Nhop

While selecting destination address of the bypass LSP, the PLR should attempt to select the router ID of the NNhop or Nhop node. If PLR and MP are in same area, then the PLR may utilize TED to determine the router ID from the interface address in RRO (if NodeID is not included in RRO). If the PLR and MP are in different IGP areas, then the PLR should use the NodeID address of NNhop MP if included in the RRO of RESV. If the NP-MP in different area has not included NodeID in RRO, then the PLR should use NP-MP’s interface address present in the RRO. The PLR should use its router ID as the source address of the bypass LSP. The PLR should also include its router ID as NodeID in PATH RRO unless configured explicitly not to include NodeID. In parallel to the attempt made to create NP-bypass or LP-bypass, the PLR initiates remote Hello to the NNhop or Nhop node respectively to track the reachability of NP-MP or LP-MP after any failure.

- If NP-bypass LSP comes up, then the PLR sets "local protection available" and "NP available" RRO flags and triggers PATH to be sent.

- If LP-bypass LSP comes up, then the PLR sets "local protection available" RRO flag and triggers PATH to be sent.

- After signaling protection availability, if the PLR finds that the protection becomes unavailable then it should attempt to make protection available. The PLR should wait for a time out before resetting RRO flags relating to protection availability and triggering PATH downstream. On the other hand, the PLR need not wait for time out to set RRO flags relating to protection availability and immediately trigger PATH downstream.

4.1.2. Remote Signaling Adjacency

A NodeID based signaling adjacency is one in which NodeID is used in source and destination address fields in RSVP Hello. [RFC4558] formalizes NodeID based Hello messages between two neighboring routers. The new procedures defined in the previous section extends the applicability of NodeID based Hello messages between two routers that may not have an interface connecting them for exchange of RSVP messages.

4.1.3. PATH RRO flags Propagation

As each node along the LSP path can make protection available, propagating PATH immediately due to change in RRO flags on any upstream node would increase control plane message load. So whenever a node receives PATH, it should check if the only change is in RRO flags. If the change is only in PATH RRO flags, then the node should decide whether to propagate the PATH based on the following rule.

- If "NP desired" flag is set and "NP available" flag has changed in Phop’s RRO flags, then PATH is triggered.

- In all other cases the change is not propagated.

4.1.4. MP Behavior

When the NNhop or Nhop node receives the triggered PATH with RRO flag(s) set, the node should check the presence of remote signaling adjacency with PLR (this check is needed to detect network being partitioned). If the flags are set and the signaling adjacency is present, the node concludes that protection has been made available at the PLR. If the PLR has included NodeID in PATH RRO, then that NodeID is the remote neighbor address. Otherwise, the PLR’s interface address in RRO will be remote neighbor address. If "NP available" flag is set by PPhop node, then it is NP-MP. Otherwise, it concludes it is LP-MP.

Once a node concludes it is MP, it should consider a "remote" state having been created from an implicit refresh directly from PLR. The "remote" state is identical to the protected LSP state except for the difference in HOP object that contains the address of remote neighbor address of node signaling adjacency with PLR. The procedures relating to "remote" state are explained in Section "Remote State Teardown". The MP should consider the "remote" state automatically deleted if:

- NP-MP receives PATH later with "NP available" flag reset in PLR’s RRO flags, or

- LP-MP receives PATH later with "local protection available" flag reset in PLR’s RRO flags, or

- Node signaling adjacency with PLR goes down, or

- MP receives backup LSP signaling from PLR overriding the shadow state, or

- MP receives PathTear, or

- MP deletes the LSP state

4.2. Impact of Failures on LSP State

4.2.1. Non-MP Behavior on Phop Link/Node Failure

When a node detects Phop link or Phop node failure and the node is not an MP, then it should send Conditional PathTear (refer to Section "Conditional PathTear" below) and delete LSP state.

4.2.2. LP-MP Behavior on Phop Link Failure

When the link to PLR fails, the link signaling adjacency to PLR will fail whereas the node signaling adjacency to PLR will remain up. So the MP should retain state.

4.2.3. LP-MP Behavior on Phop Node Failure

When the node signaling adjacency with Phop (that is also the PLR) goes down, the node should send normal PathTear and delete the LSP state.

4.2.4. NP-MP Behavior on Phop Link Failure

If the Phop link fails on NP-MP, then NP-MP should start a one shot timer (called "NodeFailureCheck" hereafter) with period greater than the hold time of NodeID neighbor session with Phop node. The purpose of "NodeFailureCheck" timer is to detect whether Phop link fails but the Phop node does not. This timer would expire or time out if the node signaling adjacency timer with Phop does not expire. If the node signaling adjacency hold time expires prior to the new timer, then the node should retain LSP state and delete the new timer. If the "NodeFailureCheck" timer expires, then the node should send Conditional PathTear and delete LSP state.

In the example topology in Figure 1, assume both A has made node protection available and C has concluded it is NP-MP. When B-C link fails then C should delete LSP state and send Conditional PathTear to D. If B has made node protection available and D has concluded it is NP-MP, then D would not delete LSP state on receiving Conditional PathTear from C. On the other hand, if D has not concluded it is NP-MP, then D would delete LSP state.

4.2.5. NP-MP Behavior on Phop Node Failure

When the Phop node fails, the node signaling adjacency with Phop will fail whereas the remote signaling adjacency to PLR will remain up. So the MP should retain state till refresh timeout.

4.2.6. NP-MP Behavior on PLR Link Failure

If the PLR link that is not attached to NP-MP fails and NP-MP receives Conditional PathTear from the Phop node, then the MP should retain state as long as the remote signaling adjacency with PLR is up. This is because the Conditional PathTear from the Phop node will not impact the "remote" state from the PLR. Note that Phop node would send Conditional PathTear if it was not an MP.

In the above example, assume C & D are NP-MP for PLRs A & B respectively. Now when A-B link fails, as B is not MP and its Phop link signaling adjacency has failed, B should delete LSP state (this behavior is required for unprotected LSPs). In the data plane, that would require B delete the label forwarding entry corresponding to the LSP. So if B’s downstream nodes C and D continue to retain state, it would not be correct for D to continue to assume itself as NP-MP for PLR B.

- As B had previously signaled NP availability, one possible solution would be to let B signal lack of NP availability before sending Conditional PathTear to C. B may trigger PATH, wait for ACK and then send Conditional PathTear to C, but this solution would increase control message load
- Or B may include both PATH with updated RRO flags and Conditional PathTear in a message bundle. While this solution would reduce control message load, the assumption that RSVP protocol could ensure two messages bundled in same message may not hold always.
- Alternatively, B may just send Conditional PathTear to C and let C interpret Conditional PathTear as implicit signaling of lack of NP availability. C should then update B’s RRO flags to signal D that node protection is longer available on B. This is the option that does not make any assumption on implementation and also not increase control message load.

The mechanism to accomplish PATH RRO update is given below.

1. B should send Conditional PathTear to C and delete LSP state.

2. When C receives Conditional PathTear, it should decide to retain LSP state as it is NP-MP of PLR A. C also should check whether Phop B had previously signaled availability of node protection. As B had previously signaled NP availability in its PATH RRO flags, C should reset "local protection available" and "NP available" on RRO flags corresponding to B and trigger PATH to D.
3. When D receives triggered PATH, it realizes that it is no longer NP-MP and so deletes the "remote" state. D does not propagate PATH further down because the only change is in PATH RRO flags of B.

4.2.7. Phop Link Failure on Node that is LP-MP and NP-MP

A node may be both LP-MP as well as NP-MP at the same time for Phop and PPhop nodes respectively. If Phop link fails on such node, the node should retain state because its Phop has made link protection available. In this scenario, "NodeFailureCheck" timer should not be started because the node would retain state irrespective of whether Phop node would fail subsequently or not.

4.2.8. Phop Node Failure on Node that is LP-MP and NP-MP

If a node that is both LP-MP and NP-MP detects Phop node failure, then the node should retain state till refresh timeout.

4.3. Conditional Path Tear

In the example provided in the previous section "NP-MP Behavior on PLR link failure", B deletes LSP state once B detects its link to Phop went down as B is not MP. If B were to send PathTear normally, then C would delete LSP state immediately. In order to avoid this, there should be some mechanism by which B could indicate to C that B does not require the receiving node to unconditionally delete the LSP state immediately. For this, B should add a new optional object in PathTear. If node C also understands the new object, then C should delete LSP state only if it is not an NP-MP - in other words C should delete LSP state if there is no "remote" PLR state on C.

4.3.1. Sending Conditional Path Tear

A node should send Conditional PathTear if the node decides to delete the LSP state under the following conditions.

- Ingress has requested node protection for the LSP, and

- PathTear is not received from upstream node, and

- A node is not a MP and Phop link or Phop node signaling adjacency goes down, or a node is an NP-MP and "NodeFailureCheck" timer started after Phop link down expires.

It should be noted that a node sends Conditional PathTear upon deleting its state in order for its Nhop node to retain state if it is NP-MP.

4.3.2. Processing Conditional Path Tear

When a node that is not an NP-MP receives Conditional PathTear, the node should delete LSP state, and process Conditional PathTear by considering it as normal PathTear. Specifically, the node should not propagate Conditional PathTear downstream but remove the optional object and send normal PathTear downstream.

When a node that is an NP-MP receives Conditional PathTear, it should not delete LSP state. The node should check whether the Phop node previously set "NP available" flag in PATH RRO flags. If the flag had been set previously by Phop, then the node should clear "local protection available" and "NP available" flags in Phop’s RRO flags and trigger PATH downstream.

If Conditional PathTear is received from a neighbor that has not advertised support (refer to Section 4.5) for the new procedures defined in this document, then the node should consider the message as normal PathTear. The node should propagate normal PathTear downstream and delete LSP state.

4.3.3. CONDITIONS object

As any implementation that does not support Conditional PathTear should ignore the new object but process the message as normal PathTear without generating any error, the Class-Num of the new object should be 10bbbbbb where ’b’ represents a bit (from Section 3.10 of [RFC2205]).

The new object is called as "CONDITIONS" object that will specify the conditions under which default processing rules of the RSVP message should be invoked.

The object has the following format:

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         Length                |  Class        |     C-type    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          Reserved                           |M|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length

This contains the size of the object in bytes and should be set to eight.

Class

TBD

C-type

1

M bit

This bit indicates that the message should be processed based on the condition whether the receiving node is Merge Point or not.

4.4. Remote State Teardown

As the refresh timeout of LSP state may be high, it is essential that LSP state be cleaned up properly even after local repair. If the Ingress intends to tear down the LSP or if PLR is unable to perform local repair, it would not be desirable to wait till backup LSP signaling to perform state cleanup. To enable LSP state cleanup when LSP is being locally repaired, nodes should send "remote" tear down message instructing the receiving node to delete LSP state.

Consider node C in above example topology (Figure 1) has gone down and B has not signaled backup LSP to D. If Ingress A intends to tear down the LSP, then the following text describes the mechanism to clean up LSP state on all nodes along the path of the LSP.

1. Ingress A sends normal PathTear to B.

2. To enable LSP state cleanup, B should send "remote" PathTear with destination IP address set to that of D, and HOP object containing local address used in remote Hello session with D.
3. On D there would be a remote signaling adjacency with B and so D should accept the remote PathTear and delete LSP state.

4.4.1. PLR Behavior on Local Repair Failure

If local repair fails on the PLR after a failure, then this should be considered as a case for cleaning up LSP state from PLR to the Egress. PLR would achieve this using "remote" PathTear to clean up state from MP. If MP has retained state, then it would propagate PathTear downstream thereby achieving state cleanup. Note that in the case of link protection, the PathTear would be directed to LP-MP node IP address rather than the Nhop interface address.

4.4.2. LSP Preemption during Local Repair

If an LSP is preempted when there is no failure along the path of the LSP, the node on which preemption occurs would send PathErr and ResvTear upstream and only delete the forwarding state. But if the LSP is being locally repaired upstream of the node on which the LSP is preempted, then the node should delete LSP state and send normal PathTear downstream. When PLR signals backup LSP, the node that was formerly MP will respond with PathErr.

4.4.2.1. Preemption after Phop Link failure

If LSP is preempted on LP-MP after its Phop or incoming link has already failed but the backup LSP has not been signaled yet, then the node should send normal PathTear and delete LSP state. As the LP-MP has retained LSP state because the PLR would refresh the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be LP-MP any more requiring the node to clean up LSP state.

4.4.2.2. Preemption after Phop Node failure

If LSP is preempted on NP-MP after its Phop node has already failed but the backup LSP has not been signaled yet, then the node should send normal PathTear and delete LSP state. As the NP-MP has retained LSP state because the PLR would refresh the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be NP-MP any more requiring the node to clean up LSP state.

Consider node B goes down on the same example topology (Figure 1). As C is NP-MP for PLR A, C should retain LSP state.

1. The LSP is preempted on C.
2. C would delete its reservation on C-D link. But C cannot send PathErr or ResvTear to PLR A because backup LSP has not been signaled yet.
3. As the only reason for C having retained state after Phop node failure was that it was NP-MP, C should send normal PathTear to D and delete LSP state. D would also delete state on receiving PathTear from C.
4. B starts backup LSP signaling to D. But as D does not have the LSP state, it should reject backup LSP PATH and send PathErr to B.
5. B should delete its reservation and send ResvTear to A.

4.5. Backward Compatibility Procedures

The "Enhanced FRR facility protection" referred below in this section refers to the set of changes that have been proposed in previous sections. Any implementation that does not support them has been termed as "existing implementation". Of the proposed extensions, signaling protection using RRO flags is expected to be backward compatible and can work safely irrespective of whether the refresh time is large. This is because the existing implementations would not send error or tear down message in response to the flags in PATH RRO but would simply ignore and propagate them. On the other hand, changes proposed relating to LSP state cleanup namely Conditional and remote PathTear require support from other nodes along the LSP path. So procedures that fall under LSP state cleanup category should be turned on only if nodes involved i.e. PLR, MP and intermediate node in the case of NP, support the extensions.

4.5.1. Detecting Support for Enhanced FRR Facility Protection

An implementation supporting the FRR facility protection extensions specified in previous sections should set a new flag "Enhanced facility protection" in CAPABILITY object in Hello messages.

- As nodes supporting the extensions should initiate Node Hellos with adjacent nodes, a node on the path of protected LSP can determine whether its Phop or Nhop neighbor supports FRR enhancements from the Hello messages sent by the neighbor.

- If a node attempts to make node protection available, then the PLR should initiate remote node signaling adjacency with NNhop. If the NNhop (a) does not reply to remote node Hello message or (b) does not set "Enhanced facility protection" flag in CAPABILITY object in the reply, then the PLR can conclude that NNhop does not support FRR extensions.

- If node protection is requested for an LSP and if (a) PPhop node has not set "local protection available" and "NP available" flags in its RRO flags or (b) PPhop node has not initiated remote node Hello messages, then the node should conclude that PLR does not support FRR extensions. The details are described in the "Procedures for backward compatibility" section below.

The new flag that will be introduced to CAPABILITY object is specified below.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num(134)|  C-Type  (1)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Reserved                      |E|T|R|S|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E bit

Indicates that the sender supports Enhanced FRR facility protection

Any node that sets the new E-bit is set in its CAPABILITY object must also set Refresh-Reduction-Capable bit in common header of all RSVP messages.

4.5.2. Procedures for backward compatibility

The procedures defined hereafter are performed on a subset of LSPs that traverse a node, rather than on all LSPs that traverse a node. This behavior is required to support backward compatibility for a subset of LSPs traversing nodes running existing implementations.

4.5.2.1. Lack of support on Downstream Node

- If the Nhop does not support enhanced facility protection FRR, then the node should reduce the "refresh period" in TIME_VALUES object carried in PATH to default small refresh default value.

- If node protection is requested and the NNhop node does not support the enhancements, then the node should reduce the "refresh period" in TIME_VALUES object carried in PATH to small refresh default value.

If the node reduces the refresh time from the above procedures, it should also not send remote PathTear or Conditional PathTear messages.

Consider the example topology in Figure 1. If C does not support scalability improvements, then:

- A and B should reduce the refresh time to default value of 30 seconds and trigger PATH

- If B is not an MP and if Phop link of B fails, B cannot send Conditional PathTear to C but should time out LSP state from A normally. This would be accomplished if A would also reduce the refresh time to default value. So if C does not support enhanced facility protection, then Phop B and PPhop A should reduce refresh time to small default value.

4.5.2.2. Lack of support on Upstream Node

- If Phop node does not support enhanced facility protection, then the node should reduce the "refresh period" in TIME_VALUES object carried in RESV to default small refresh time value.

- If node protection is requested and the Phop node does not support the enhancements, then the node should reduce the "refresh period" in TIME_VALUES object carried in PATH to default value.

- If node protection is requested and PPhop node does not support the enhancements, then the node should reduce the "refresh period" in TIME_VALUES object carried in RESV to default value.

- If the node reduces the refresh time from the above procedures, it should also not execute MP determination procedures.

5. Security Considerations

This document does not introduce new security issues. The security considerations pertaining to the original RSVP protocol [RFC2205] remain relevant.

6. IANA Considerations

TBD

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4090] Pan, P., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

[RFC2961] Berger, L., "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[RFC3209] Awduche, D., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC2205] Braden, R., "Resource Reservation Protocol (RSVP)", RFC 2205, September 1997.

[RFC4558] Ali, Z., "Node-ID Based Resource Reservation (RSVP) Hello: A Clarification Statement", RFC 4558, June 2006.

8. Acknowledgments

Thanks to Raveendra Torvi and Yimin Shen for their comments and inputs.

9. Authors’ Addresses

Chandra Ramachandran
Juniper Networks
csekar@juniper.net

Yakov Rekhter
Juniper Networks
Email: yakov@juniper.net

Markus Jork
Juniper Networks
Email: mjork@juniper.net

Contributors

Harish Sitaraman
Juniper Networks
Email: hsitaraman@juniper.net

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 16, 2015

X. Chen
Z. Li
Huawei Technologies

August 15, 2014

Yang Model for MPLS LDP
draft-chen-mpls-ldp-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage MPLS LDP.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF[RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces(e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and Programmatic APIs.

This document defines a YANG data model that can be used to configure and manage MPLS LDP. It includes the core LDP[RFC5036] and mLDP[RFC6388]. In addition, features described in different separate MPLS LDP RFC are also supported.

2. Terminology

LDP: Label Distribution Protocol

mLDP: Multipoint extensions for LDP

GR: Graceful Restart

GTSM: Generalized TTL Security Mechanism

3. Design of Data Model

3.1. Overview

The LDP Yang module is divided into two main containers :

o ldpGlobalPara : that contains global writable configuration objects.

o ldpInstances : that contains per-instance writable configuration objects.

The figure below describes the overall structure of the LDP Yang module :

```
module: mplsldp
   +--rw mplsLdp
      +--rw ldpGlobalPara
      |  +--rw grEnable?                       boolean
      |  +--rw reconnectTime?                  uint32
      |  +--rw recoveryTime?                   uint32
      |  +--rw peerLiveTime?                   uint32
      |  +--rw backOffInitTime?                uint32
      |  +--rw backOffMaxTime?                 uint32
      +--rw ldpInstances
         +--rw ldpInstance* [vrfName]
            +--rw vrfName              string
            +--rw lsrid                inet:ipv4-address
            +--rw igpSyncDelayTime?   uint32
            +--rw ldpInterfaces
            |  ...
            +--rw ldpRemotePeers
            |  ...
            +--rw ldpPeers
            |  ...
            +--rw ldpGtsms
            |  ...
            +--rw mldpP2mpTunnels
            |  ...
            +--rw mldpP2mpLeafLsps
               ...
```

3.2. LDP Global Configuration

LDP global configuration container includes the global parameters such as graceful restart, backoff timer, ...

3.3. LDP Per-instance Configuration

LDP per-instance configuration container includes parameters of the public LDP instance or the LDP instance binding a specific VRF. LDP per-instance configuration container is divided into:

o Per-instance parameters.

o Per-interface configuration of the LDP instance

o Remote peer configuration of the LDP instance

o Peer configuration of the LDP instance

o GTSM[RFC6720] configuration of the LDP instance

o mLDP P2MP tunnel configuration of the LDP instance

o mLDP P2MP Leaf LSP configuration of the LDP instance

3.3.1. Per-instance Parameters

The per-instance parameter includes the name of the VRF bound by the LDP instance, LSR ID and timer parameters for LDP IGP Synchronization[RFC5443].

3.3.2. Per-interface Configuration of LDP Instance

Per-interface configuration of the LDP instance includes the interface name, hello timer parameters, keepalive timer parameters, timer parameters for IGP LDP synchronization, transport address.

```
            +--rw ldpInterfaces
            |  +--rw ldpInterface* [ifName]
            |     +--rw ifName                     ifName
            |     +--rw helloSendTime?             uint16
            |     +--rw helloHoldTime?             uint16
            |     +--rw keepaliveSendTime?         uint16
            |     +--rw keepaliveHoldTime?         uint16
            |     +--rw igpSyncDelayTime?          uint32
            |     +--rw transportAddrInterface?    ifName
```

3.3.3. Remote Peer Configuration of LDP Instance

The remote peer configuration of the LDP instance includes the name and the remote IP address of the remote peer, hello timer parameters, keepalive timer parameters, timer parameters for IGP LDP synchronization,

```
            +--rw ldpRemotePeers
            |  +--rw ldpRemotePeer* [remotePeerName]
            |     +--rw remotePeerName            string
            |     +--rw remoteIp?                 inet:ipv4-address
            |     +--rw helloSendTime?            uint16
            |     +--rw helloHoldTime?            uint16
            |     +--rw keepaliveSendTime?        uint16
            |     +--rw keepaliveHoldTime?        uint16
            |     +--rw igpSyncDelayTime?         uint32
```

3.3.4. Peer Configuration of LDP Instance

The peer configuration of the LDP instance includes the peer IP address, authentification parameters for the peer. It may also include the policy to control the distribution of label mapping over the peer which will be defined in the future version.

```
            +--rw ldpPeers
            |  +--rw ldpPeer* [peerid]
            |     +--rw peerid         inet:ipv4-address
            |     +--rw md5Password?   string
```

3.3.5. GTSM Configuration of LDP Instance

The GTSM configuration includes the peer transport address and hop limit.

```
            +--rw ldpGtsms
               +--rw ldpGtsm* [peerTransportAddr]
                  +--rw peerTransportAddr    inet:ipv4-address
                  +--rw gtsmHops             uint16
```

3.3.6. mLDP P2MP Tunnel Configuration of LDP Instance

In the ingress LSR, mLDP P2MP tunnel should be defined as the entity used for multicast traffic carried by the mLDP P2MP tunnel. The mLDP P2MP tunnel configuration includes the tunnel name, root IP address and the P2MP LSP ID.

```
               +--rw mldpP2mpTunnels
               |  +--rw mldpP2mpTunnel* [tunnelName]
               |     +--rw tunnelName    string
               |     +--rw rootIp?       inet:ipv4-address
               |     +--rw p2mpLspId?    uint32
```

3.3.7. mLDP P2MP Leaf LSP Configuration of LDP Instance

mLDP P2MP leaf LSP configuration includes the P2MP LSP name, root IP address and P2MP LSP ID.

```
               +--rw mldpP2mpLeafLsps
                  +--rw mldpP2mpLeafLsp* [p2mpLspName]
                     +--rw p2mpLspName    string
                     +--rw rootIp?        inet:ipv4-address
                     +--rw p2mpLspId?     uint32
```

4. LDP Yang Module

```
module mplsldp {
    namespace "urn:huawei:params:xml:ns:yang:mplsldp";
    // replace with IANA namespace when assigned - urn:ietf:params:xml:ns:yang:1
    prefix "mplsldp";
  import ietf-inet-types {
    prefix inet;
  }
    organization
      "Huawei Technologies Co., Ltd.";
    contact
      "jescia.chenxia@huawei.com
       lizhenbin@huawei.com";
    description
      "This YANG module defines the generic configuration
     data for LDP, which is common across all of the vendor
     implementations of the protocol. It is intended that the module
     will be extended by vendors to define vendor-specific
     LDP configuration parameters.";
  revision 2014-08-16 {
    description
      "Initial revision.";
  }

  typedef ifName {
        description "ifName is like ethernet1/1/1/1";
        type string {
            length "1..63";
        }
    }
    container mplsLdp {

        container ldpGlobalPara {

            leaf grEnable {
                description
                  "Specifies the LDP GR capability.
                   Enabling or disabling of GR will cause the re-establishment o
f all LDP sessions.";
                config "true";
                type boolean;
                default "false";
            }
            leaf reconnectTime {
                description
                  "Specifies the value of the LDP session reconnection timer.
                   The value is in seconds.";
                config "true";
                default "300";
                type uint32 {
                    range "3..3600";
                }
            }
            leaf recoveryTime {
                description
                  "Specifies the value of the LSP recovery timer (s).
                   The value is in seconds.";
                config "true";
                default "300";
                type uint32 {
                    range "3..3600";
                }
            }
            leaf peerLiveTime {
                description
                  "Specifies the value of the neighbor keeplive timer (s).
                   The value is in seconds.";
                config "true";
                default "600";
                type uint32 {
                    range "3..3600";
                }
            }
            leaf backOffInitTime {
                description
                  "Specifies the init value of the exponential backoff timer (s)
.
                   The value is in seconds.";
                config "true";
                default "15";
                type uint32 {
                    range "5..2147483";
                }
            }
            leaf backOffMaxTime {
                description
                  "Specifies the maximum value of the exponential backoff Timer(
s).
                   The value is in seconds.";
                config "true";
                default "120";
                type uint32 {
                    range "5..2147483";
                }
            }
        }

        container ldpInstances {

            list ldpInstance {

                key "vrfName";
                max-elements "unbounded";
                min-elements "0";
                description "Specifies a list of LDP instances.";

                leaf vrfName {
                    description
                      "Name of an LDP instance.
                      If the name string is empty the instance means a public in
stance whose name is _public_.";
                    config "true";
                    //default "_public_";
                    type string {
                        length "0..32";
                    }
                }
                leaf lsrid {
                    description "LSR ID of an instance.";
                    config "true";
                    mandatory "true";
                    type inet:ipv4-address;
                }
                leaf igpSyncDelayTime {
                    description
                      "Specifies the interval at which an interface waits to est
ablish an LSP after an LDP session is set up.
                       The value is in seconds.";
                    config "true";
                    default "10";
                    type uint32 {
                        range "0..65535";
                    }
                }
                container ldpInterfaces {

                    list ldpInterface {

                        key "ifName";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies an LDP interface.";

                        leaf ifName {
                            description "Interface name.";
                            config "true";
                            type ifName;
                        }
                        leaf helloSendTime {
                            description
                              "Specifies the value of the Hello packet sending t
imer.

                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf helloHoldTime {
                            description
                              "Specifies the interval value of the Hello hold ti
mer.

                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "3..65535";
                            }
                        }
                        leaf keepaliveSendTime {
                            description
                              "Specifies the value of the Keepalive packet sendi
ng timer (s).

                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf keepaliveHoldTime {
                            description
                              "Specifies the value of the Keepalive packet holdi
ng timer (s).

                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "30..65535";
                            }
                        }
                        leaf igpSyncDelayTime {
                            description
                              "Specifies an interval at which an interface waits
 to establish an LSP after an LDP session is set up.
                               The value is in seconds.";
                            config "true";
                            type uint32 {
                                range "0..65535";
                            }
                        }
                        leaf transportAddrInterface {
                            description "Configures an interface address to be u
sed as the transport address.";
                            config "true";
                            type ifName;
                        }

                    }

                }

                container ldpRemotePeers {

                    list ldpRemotePeer {

                        key "remotePeerName";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies a remote LDP neighbor.";

                        leaf remotePeerName {
                            description "Specifies the name of a remote neighbor
.";

                            config "true";
                            type string {
                                length "1..32";
                            }
                        }
                        leaf remoteIp {
                            description "Specifies the IPv4 address of a remote
neighbor.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf helloSendTime {
                            description
                              "Specifies the value of the Hello packet sending t
imer.
```

```
                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf helloHoldTime {
                            description
                              "Specifies the value of the Hello packet holding t
imer (s).
                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "3..65535";
                            }
                        }
                        leaf keepaliveSendTime {
                            description
                              "Specifies the value of the Keepalive packet sendi
ng timer.
                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf keepaliveHoldTime {
                            description
                              "Specifies the value of the Keepalive holding time
r.
                               The value is in seconds.";
                            config "true";
                            type uint16 {
                                range "30..65535";
                            }
                        }
                        leaf igpSyncDelayTime {
                            description
                              "Specifies an interval at which an interface waits
 to establish an LSP after an LDP session is set up.
                               The value is in seconds.";
                            config "true";
                            type uint32 {
                                range "0..65535";
                            }
                        }

                    }

                }

                container ldpPeers {
                    list ldpPeer {

                        key "peerid";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies an LDP peer.";

                        leaf peerid {
                            description "Specifies an LDP peer ID.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf md5Password {
                            description "Specifies an MD5 password.";
                            config "true";
                            type string {
                      length "1..255";
                  }
                        }

                    }

                }

                container ldpGtsms {

                    list ldpGtsm {

                        key "peerTransportAddr";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies a GTSM security attribute.";

                        leaf peerTransportAddr {
                            description "Specifies a GTSM transport address.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf gtsmHops {
                            description "Specifies the maximum number of GTSM ho
ps. The value is an integer ranging from 1 to 255.";
                            config "true";
                            mandatory "true";
                            type uint16 {
                                range "1..255";
                            }
                        }
                    }
                }

                container mldpP2mpTunnels {

                    list mldpP2mpTunnel {

                        key "tunnelName";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies the mldp p2mp lsp configuration i
n the root node.";

                        leaf tunnelName {
                            description "Mldp p2mp tunnel name in root node whic
h can be used by service.";
                            config "true";
                            type string {
                                length "1..31";
                            }
                        }
                        leaf rootIp {
                            description "Specifies the root ip address of mldp p
2mp lsp.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf p2mpLspId {
                            description "Specifies the LSP ID of mldp p2mp lsp i
f the generic LSP identifier is a type of opaque value element.";
                            config "true";
                            type uint32 {
                                range "1..8192";
                            }
                        }
                    }

                }

                container mldpP2mpLeafLsps {

                    list mldpP2mpLeafLsp {

                        key "p2mpLspName";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies the mldp p2mp lsp configuration i
n the leaf node.";

                        leaf p2mpLspName {
                            description "The name of mldp p2mp lsp configuration
 in the leaf node.";
                            config "true";
                            type string {
                                length "1..31";
                            }
                        }
                        leaf rootIp {
                            description "Specifies the root ip address of mldp p
2mp lsp.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf p2mpLspId {
                            description "Specifies the LSP ID of mldp p2mp lsp i
f the generic LSP identifier is a type of opaque value element.";
                            config "true";
                            type uint32 {
                                range "1..8192";
                            }
                        }
                    }

                }

            }

        }

    }

}
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

The data model defined does not create any security implications. This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

7. Acknowledgements

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, March 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6388] Wijnands, IJ., Minei, I., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, November 2011.

[RFC6720] Pignataro, C. and R. Asati, "The Generalized TTL Security Mechanism (GTSM) for the Label Distribution Protocol (LDP)", RFC 6720, August 2012.

Authors’ Addresses

Xia Chen
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: jescia.chenxia@huawei.com

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

Email: lizhenbin@huawei.com

Network Working Group M. Chen
Internet-Draft X. Xu
Intended status: Standards Track Z. Li
Expires: April 16, 2015 Huawei
L. Fang
Microsoft
G. Mirsky
Ericsson
October 13, 2014

MultiProtocol Label Switching (MPLS) Source Label
draft-chen-mpls-source-label-06

Abstract

A MultiProtocol Label Switching (MPLS) label was originally defined to identify a Forwarding Equivalence Class (FEC). A packet is assigned to a specific FEC based on its network layer destination address, and optionally Class of Service. It’s difficult or even impossible to derive the source identity information from the label. For some applications, source identification is a critical requirement. For example, performance monitoring, where the monitoring node needs to identify where a packet was sent from.

This document introduces the concept of Source Identifier (SI) that identifies the ingress Label Switching Router (LSR) of a Label Switched Path (LSP). A SI is unique within a domain that is referred to as Source Identifier Administrative Domain (SIAD).

This document also introduces the concept of Source Label (SL) that is carried in the label stack and carries the SI of the ingress LSR of an LSP. Source Label is preceded by a Source Label Indicator (SLI) when included the label stack and is not used for forwarding.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Problem Statement and Introduction

A MultiProtocol Label Switching (MPLS) label [RFC3031] was originally defined for packet forwarding and assumes the forwarding/destination address semantics. As no source identity information is carried in the label stack, in many cases there is no way to directly derive the source identity information from the label or label stack.

MPLS LSPs can be categorized into four different types:

o Point-to-Point (P2P)

o Point-to-Multipoint (P2MP)

o Multipoint-to-Point (MP2P)

o Multipoint-to-Multipoint (MP2MP)

For P2P and P2MP LSPs (e.g., the Resource Reservation Protocol Traffic Engineering (RSVP-TE) [RFC3209] based and statically configured P2P and P2MP LSPs), the source identity may be implicitly derived by the egress LSR from the label when Penultimate Hop Popping (PHP) is disabled and the correlation between ingress LSR and the LSP is explicitly signaled through the control plane. Such LSP may be characterized as MPLS-TP LSP [RFC5960].

However, for MP2P and MP2MP LSPs (e.g., the Label Distribution Protocol (LDP) based LSPs [RFC5036] [RFC6388], and Layer 3 Private Network (L3VPN) [RFC4364] LSPs), ingress LSRs of those LSPs cannot be identified by egress LSRs.

Comparing to the pure IP forwarding where both source and destination addresses are encoded in the IP packet header, the essential issue of the MPLS encoding is that the label stack does not explicitly include any source identity information. For some applications, source identification is a critical requirement. For example, performance monitoring, the monitoring nodes need to identify where packets were sent from and then can count the packets according to some constraints.

This document introduces the concept of Source Label (SL). An SL is carried in the label stack and carries the identifier of the ingress LSR that originated the MPLS frame.

2. Terminology

SI - Source Identifier

SIAD - Source Identifier Administrative Domain

SL - Source Label

SLC - Source Label Capability

SLI - Source Label Indicator

3. Source Label

A Source Label is defined to carry an identifier (Source Identifier) of a node that is (one of) the ingress LSR(s) to specific LSP. Source Label SHOULD NOT be used for forwarding and is not signaled.

A Source Identifier (SI) is a number in the range of [16, 65535]. Each node in a domain MUST be allocated one or more unique SIs, the domain is referred as a "Source Identifier Administrative Domain" (SIAD). For most of the use cases, one SI per LSR would be sufficient. But for some cases, there may be need for more than one SIs. For example, in the L3VPN scenario, it may be necessary to allocate a dedicated SI to identify each VPN instance.

In order to indicate whether a label is a Source Label, a Source Label Indicator (SLI) is introduced. The SLI is a special purpose label [RFC7274] that is placed immediately before the source label in the label stack, which is used to indicate that the next label in the label stack is the Source Label. The value of SLI is TBD1. The SL is an example of context label [RFC5331], the SLI is the context.

To prevent the Source Label from leaking to unintended domains, two aspects need to be considered:

o In the control plane, the Source Label MUST NOT be distributed outside the SIAD where it is used. Since the ingress LSR is based on the Source Label Capability signaled by the egress LSR to determine whether to insert the Source Label, the SLC signaling MUST make sure that the SLC will not be signaled to the LSRs that reside in other SIADs.

o In the data plane, the domain boundary nodes (e.g., the ASBR) SHOULD have the capability to filter out the packets that carry the SL/SLI and are received from other SIADs. For example, some policies (e.g., using ACL) could be deployed at the ASBR to filter out the packets that carry SL/SLI and are from other SIADs.

4. Performance Measurement Use Case

There are two general types of performance measurement: one is active performance measurement, and the other is passive performance measurement.

In active performance measurement the receiver measures the injected packets to evaluate the performance of a path. The active measurement measures the performance of the extra injected packets. The IP Performance Metrics (IPPM) working group has defined specifications [RFC4656][RFC5357] for active performance measurement.

In passive performance measurement, no additional traffic is injected into the flow and measurements are taken to record the performance metrics of the data traffic. The MPLS performance measurement protocol [RFC6374] for packet loss is an example of passive performance measurement, but currently it can only be measured for MPLS-TE LSPs. For a specific receiver, in order to count the received packets of a flow, the system doing the measurement (e.g., egress router) needs to know which target flow a received packet belongs to. Source identification is therefore necessary. Source identification may be achieved by including appropriate MEP-ID [RFC6428].

As discussed in the previous section, the existing MPLS label or label stack does not carry the source information. So, for an LSP, the ingress LSR can put its SI in the Source Label, and then the egress LSR can use the SI to identify the packet’s source, in order to facilitate accounting.

5. Data Plane Processing

5.1. Ingress LSR

For an LSP, the ingress LSR MUST make sure that the egress LSR is able to process the Source Label before inserting the SLI/SL combination into the label stack. Therefore, an egress LSR SHOULD signal (see Section 6) to the ingress LSR whether it is able to process the Source Label. Once the ingress LSR knows that the egress LSR can process Source Label, it can choose whether or not to insert the SL and SLI into the label stack.

When an SL to be included in a label stack, the steps are as follows:

1. Push the SL, the TTL of the SL MUST be set to 1, the BoS bit for the SL depends on whether the SL is the bottom label. Setting and interpretation of TC field of the SL is for further study;

2. Push the SLI, the TTL and TC fields for the SLI MUST be set to the same values as for the LSP Label (L);

3. Push the LSP Label (L).

Then the label stack looks like: <...L, SLI, SL [,...]>. There MAY be multiple combinations of SLI and SL inserted into the label stack, each combination is related to an LSP. For the given LSP, only one combination of SLI and SL MUST be inserted.

5.2. Transit LSR

There is no change in forwarding behavior for transit LSRs. If a transit LSR can recognize the SLI, it can use the SL to collect traffic throughput and/or measure the performance of the LSP.

5.3. Egress LSR

When an egress LSR receives a packet with a SLI/SL combination, if the egress LSR is able to process the SL; it pops the LSP label (if any), SLI and SL; then processes remaining packet header as normal. If the egress LSR is not able to process the SLI, the packet SHOULD be dropped as specified for the handling of any unknown label according to [RFC3031].

5.4. Penultimate Hop LSR

There is no change in forwarding behavior for the penultimate hop LSR.

6. Source Label Capability Signaling

Before inserting a Source Label in the label stack, an ingress LSR SHOULD know whether the egress LSR is able to process the SLI and SL. Therefore, an egress LSR SHOULD signal to the ingress LSRs its ability to process the SLI and SL. This is called Source Label Capability (SLC), it is very similar to the "Entropy Label Capability (ELC)"[RFC6790].

6.1. LDP Extensions

A new LDP TLV [RFC5036], SLC TLV, is defined to signal an egress’s ability to process Source Label. The SLC TLV MAY appear as an Optional Parameter of the Label Mapping Message. The presence of the SLC TLV in a Label Mapping Message indicates to ingress LSRs that the egress LSR can process Source Labels for the associated LSP.

The structure of the SLC TLV is shown below.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |U|F|       Type (TBD2)       |           Length (0)          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Source Label Capability TLV

This U bit MUST be set to 1. If the SLC TLV is not understood by the receiver, then it MUST be ignored.

This F bit MUST be set to 1. Since the SLC TLV is going to be propagated hop-by-hop, it should be forwarded even by nodes that may not understand it.

Type: TBD2.

Length field: This field specifies the total length in octets of the SLC TLV and is defined to be 0.

An LSR that receives a Label Mapping with the SLC TLV but does not understand it MUST propagate it intact to its neighbors and MUST NOT send a notification to the sender (following the meaning of the U- and F-bits). If the LSR has no other neighbors and does not understand the SLC TLV, means it is the ingress LSR, it could just ignore it. An LSR X may receive multiple Label Mappings for a given FEC F from its neighbors. In its turn, X may advertise a Label Mapping for F to its neighbors. If X understands the SLC TLV, and if any of the advertisements it received for FEC F does not include the SLC TLV, X MUST NOT include the SLC TLV in its own advertisements of F. If all the advertised Mappings for F include the SLC TLV, then X MUST advertise its Mapping for F with the SLC TLV. If any of X’s neighbors resends its Mapping, sends a new Mapping or sends a Label Withdraw for a previously advertised Mapping for F, X MUST re-evaluate the status of SLC for FEC F, and, if there is a change, X MUST re-advertise its Mapping for F with the updated status of SLC.

LDP is normally running within an AS, technically, it can be deployed across ASes. An implementation supports the SLC MUST support a per-session/per-interface configuration item to enable/disable the SLC. For the session/interface that connects to other SLADs, the SLC MUST be disabled.

6.2. BGP Extensions

When Border Gateway Protocol (BGP) [RFC4271] is used for distributing Network Layer Reachability Information (NLRI) as described in, for example, [RFC3107], [RFC4364], the BGP UPDATE message may include the

SLC attribute as part of the Path Attributes. This is an optional, non-transitive BGP attribute of value TBD3. The inclusion of this attribute with an NLRI indicates that the advertising BGP router can process Source Labels as an egress LSR for all routes in that NLRI.

A BGP speaker S that originates an UPDATE should include the SLC attribute only if both of the following are true:

A1: S sets the BGP NEXT_HOP attribute to itself AND

A2: S can process source labels.

Suppose a BGP speaker T receives an UPDATE U with the SLC attribute. T has two choices. T can simply re-advertise U with the SLC attribute if either of the following is true:

B1: T does not change the NEXT_HOP attribute OR

B2: T simply swaps labels without popping the entire label stack and processing the payload below.

An example of the use of B1 is Route Reflectors. However, if T changes the NEXT_HOP attribute for U and in the data plane pops the entire label stack to process the payload, T MAY include an SLC attribute for UPDATE U’ if both of the following are true:

C1: T sets the NEXT_HOP attribute of U’ to itself AND

C2: T can process source labels. Otherwise, T MUST remove the SLC attribute.

6.2.1. Sending/Receiving Restriction

An implementation that supports the SLC MUST support per-session configuration item, SL_SESSION, that indicates whether the SLC is enabled or disabled for use on that session.

- The default value of SL_SESSION, for EBGP sessions, MUST be "disabled".

- The default value of SL_SESSION, for IBGP and confederation-EBGP [RFC5065]sessions, SHOULD be "enabled."

The SLC attribute MUST NOT be sent on any BGP session for which SL_SESSION is disabled.

If an SLC attribute is received on a BGP session for which SL_SESSION is disabled, the attribute MUST be treated exactly as if it were an unrecognized non-transitive attribute. That is, "it MUST be quietly ignored and not passed along to other BGP peers" (see [RFC4271], section 5).

6.3. IGP Extensions

IGP based SLC applies to the scenarios where IGP is used for label mapping (e.g., Segment Routing). IGP SLC signaling is defined in [I-D.chen-isis-source-identifier-distribution] and [I-D.chen-ospf-source-identifier-distribution], the presence of a Source Identifier TLV/sub-TLV MUST be interpreted as support of SLC by the LSR. That means the SLC is implicitly indicated by receiving a SI distribution from an LSR.

7. Source Identifier Distribution

Based on the Source Identifier, an egress or intermediate LSR can identify from where an MPLS packet is sent. To achieve this, the egress and/or intermediate LSRs have to know which ingress LSR is related to which Source Identifier before using the Source Identifier to derive the source information. Therefore, there needs to be a mechanism to distribute the mapping information between an ingress LSR and its SI(s).

IGP based SI distribution documents, [I-D.chen-isis-source-identifier-distribution], [I-D.chen-ospf-source-identifier-distribution], define extensions to corresponding IGP protocols necessary for intra-AS scenario.

For inter-AS scenario, BGP extension is a naturally choice and can be used to convey SI mapping information from one AS to other ASes. The BGP extension draft is work in progress. For BGP based SI distribution, it requires that SIs MUST not be sent out of a SIAD. The similar sending and receiving restriction as defined in Section 6.2.1 is also needed.

8. IANA Considerations

8.1. Source Label Indication

IANA is required to allocate a special purpose label (TBD1) for the Source Label Indicator (SLI) from the "Multiprotocol Label Switching Architecture (MPLS) Label Values" Registry.

8.2. LDP Source Label Capability TLV

IANA is requested to allocate a value of TBD2 from the IETF Consensus range (0x0001-0x07FF) in the "TLV Type Name Space" registry as the "Source Label Capability TLV".

8.3. BGP Source Label Capability Attribute

IANA is requested to allocate a Path Attribute Type Code TBD3 from the "BGP Path Attributes" registry as the "BGP Source Label Capability Attribute".

9. Security Considerations

This document introduces the SIAD that is the scope of a SL. The SLC and SI MUST NOT be signaled and distributed outside one SIAD. BGP based SLC and SI distribution is controlled by SL_SESSION configuration. Improper configuration on both ends of an EBGP connection could result in the SLC and SI being passed from one SIAD to another. This would likely result in potential SI conflicts.

To prevent packets carrying SL/SLI from leaking from one SIAD to another, the SIAD boundary nodes SHOULD deploy some policies (e.g., ACL) to filter out the packets. Specifically, in the sending end, the SIAD boundary node SHOULD filter out the packets that carry the SLI and are sent to other SIADs; in the receiving end, the SIAD boundary node SHOULD drop the packets that carry the SLI and are from other SIADs.

10. Acknowledgements

The process of "Source Label Capability Signaling" is largely referred to the process of "ELC signaling"[RFC6790].

The authors would like to thank Carlos Pignataro, Loa Andersson , Curtis Villamizar, Eric Osborne, Eric Rosen, Yimin Shen, Lizhong Jin, Ross Callon and Yakov Rekhter for their review, suggestion and comments to this document.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5420] Farrel, A., Papadimitriou, D., Vasseur, JP., and A. Ayyangarps, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)", RFC 5420, February 2009.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, September 2011.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, June 2014.

11.2. Informative References

[I-D.chen-isis-source-identifier-distribution] Chen, M. and G. Mirsky, "Extensions to ISIS for Source Identifier Distribution", draft-chen-isis-source-identifier-distribution-00 (work in progress), October 2014.

[I-D.chen-ospf-source-identifier-distribution] Chen, M. and G. Mirsky, "Extensions to OSPF for Source Identifier Distribution", draft-chen-ospf-source-identifier-distribution-00 (work in progress), October 2014.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC4761] Kompella, K. and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, January 2007.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, August 2008.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5960] Frost, D., Bryant, S., and M. Bocci, "MPLS Transport Profile Data Plane Architecture", RFC 5960, August 2010.

[RFC6388] Wijnands, IJ., Minei, I., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, November 2011.

[RFC6428] Allan, D., Swallow Ed. , G., and J. Drake Ed. , "Proactive Connectivity Verification, Continuity Check, and Remote Defect Indication for the MPLS Transport Profile", RFC 6428, November 2011.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

Authors’ Addresses

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Luyuan Fang
Microsoft

Email: lufang@microsoft.com

Greg Mirsky
Ericsson

Email: Gregory.mirsky@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 16, 2015

X. Chen
Z. Li
X. Zeng
Huawei Technologies

August 15, 2014

Yang Model for MPLS Traffic Engineering(TE)
draft-chen-mpls-te-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage MPLS TE.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF[RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces(e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage MPLS TE. Both P2P TE and P2MP TE are supported.

2. Terminology

TE: Traffic Engineering

P2MP TE: Point-to-Multipoint Traffic Engineering

FRR: Fast Re-Route

SRLG: Shared Risk Link Group

CSPF: Constrained Shortest Path First

3. Design of Data Model

3.1. Overview

The MPLS TE Yang module is divided in for main containers :

o mplsTeSite: that contains global writable configuration objects for MPLS TE.

o teLinks: that contains writable configuration objects for MPLS TE link.

o explicitPaths : that contains writable configuration objects for explicit path used for MPLS TE tunnel.

o p2mpLeafLists : that contains writable configuration objects for the list of leaf nodes of P2MP TE tunnel.

o rsvpTeTunnels : that contains writable configuration objects for RSVP-TE tunnels.

o rsvpTESite : that contains writable configuration objects for RSVP-TE.

o rsvpInterfaces : that contains writable configuration objects for RSVP-TE.

o cspfCfg : that contains writable configuration objects for CSPF.

o p2mpTeTemplates: that contains writable configuration objects for P2MP TE tunnel template which is used for setting up P2MP TE tunnel triggered by multicast services.

When implement MPLS TE features, it is necessary to flood MPLS TE link information through IGP signaling including OSPF-TE and ISIS-TE. The Yang configuration of ISIS-TE and OSPF-TE is out of scope of this document.

The figure below describe the overall structure of the MPLS TE Yang module :

```
   module: mplste
      +--rw mplsTe
         +--rw mplsTeSite
         |  ...
         +--rw teLinks
         |  ...
         +--rw explicitPaths
         |  ...
         +--rw p2mpLeafLists
         |  ...
         +--rw rsvpTeTunnels
         |  ...
         +--rw rsvpTESite
         |  ...
         +--rw rsvpInterfaces
         |  ...
         +--rw cspfCfg
         |  ...
         +--rw p2mpTeTemplates
            ...
```

3.2. MPLS TE Global Configuration

MPLS TE global configuration includes global parameters to control the flooding the TE link information, LSP switch auto FRR, auto bandwidth adjustment.

```
         +--rw mplsTeSite
         |  +--rw enablePeriFl?       boolean
         |  +--rw floodingInterval?   uint32
         |  +--rw switchDelay?        uint32
         |  +--rw autoFrrEnable?      boolean
         |  +--rw autobwEnable?       boolean
         |  +--rw autobwInterval?     uint32
```

3.3. MPLS TE Link Configuration

MPLS TE link configuration includes the parameters of the MPLS links such as bandwidth, administration group, SRLG, TE metric, auto FRR mode and the threshold value to control the flooding of MPLS TE link information.

```
         +--rw teLinks
         |  +--rw teLink* [interfaceName]
         |     +--rw interfaceName                 ifName
         |     +--rw teIfMaxreservablebandwidth?   uint32
         |     +--rw teIfBc0bandwidth?             uint32
         |     +--rw adminGroups?                  string
         |     +--rw srlgGroups
         |     |  +--rw srlgGroup* [teIfSrlgValue]
         |     |     +--rw teIfSrlgValue    uint32
         |     +--rw bwChangeThresholdDown?        uint32
         |     +--rw bwChangeThresholdUp?          uint32
         |     +--rw teIfMetric?                   uint32
         |     +--rw autoFrrMode?                  enumeration
```

3.4. Explicit Path Configuration

Explicit path configuration includes the list of IP addresses of strict or loose hops of the explicit path.

```
         +--rw explicitPaths
         |  +--rw explicitPath* [explicitPathName]
         |     +--rw explicitPathName    string
         |     +--rw explicitPathHops
         |        +--rw explicitPathHop* [mplsTunnelHopIndex]
         |           +--rw mplsTunnelHopIndex       uint32
         |           +--rw mplsTunnelHopIpAddr      inet:ipv4-address
         |           +--rw mplsTunnelHopType?       enumeration
         |           +--rw mplsTunnelHopAddrType?   enumeration
```

3.5. P2MP TE Leaf List Configuration

P2MP TE leaf list configurations includes the list of leaf nodes of the P2MP TE tunnel and the explicit paths used for these leaf nodes.

```
         +--rw p2mpLeafLists
         |  +--rw p2mpLeafList* [leafListName]
         |     +--rw leafListName    string
         |     +--rw leafs
         |        +--rw leaf* [leafIpAddr]
         |           +--rw leafIpAddr          inet:ipv4-address
         |           +--rw explicitPathName?   string
```

3.6. RSVP-TE Tunnel Configuration

RSVP-TE tunnel configuration includes the parameters for the RSVP-TE tunnel and the parameters for different LSPs (primary LSP, hot-standby LSP, ordinary backup LSP and best-effort LSP) of the RSVP-TE tunnels.

```
         +--rw rsvpTeTunnels
         |  +--rw rsvpTeTunnel* [tunnelName]
         |     +--rw tunnelName                   string
         |     +--ro mplsTunnelIngressLSRId?      inet:ipv4-address
         |     +--rw mplsTunnelEgressLSRId?       inet:ipv4-address
         |     +--rw mplsTunnelIndex?             uint16
         |     +--rw mplsTunnelBandwidth?         uint32
         |     +--rw mplsTeTunnelSetupPriority?   uint8
         |     +--rw holdPriority?                uint8
         |     +--rw hotStandbyEnable?            boolean
         |     +--rw hsbRevertiveMode?            enumeration
         |     +--rw hotStandbyWtr?               uint32
         |     +--rw ordinaryEnable?              boolean
         |     +--rw bestEffortEnable?            boolean
         |     +--rw disableCspf?                 boolean
         |     +--rw tunnelPaths
         |     |  +--rw tunnelPath* [pathType]
         |     |     +--rw pathType            enumeration
         |     |     +--rw explicitPathName?   string
         |     |     +--ro includeAll?         string
         |     |     +--rw includeAny?         string
         |     |     +--rw excludeAny?         string
         |     |     +--rw hopLimit?           uint32
         |     +--rw resvStyle?                   enumeration
         |     +--rw mplsTunnelRecordRoute?       enumeration
         |     +--rw reoptimization?              boolean
         |     +--rw reoptiFrequency?             uint32
         |     +--rw tieBreaking?                 enumeration
         |     +--rw pathMetricType?              enumeration
         |     +--rw AutoBandwidths
         |     |  +--rw AutoBandwidth
         |     |     +--rw AutoBwMode?      enumeration
         |     |     +--rw thresholdPerc?   uint32
         |     |     +--rw AutoBwFreq?      uint32
         |     |     +--rw AutoBwMax?       uint32
         |     |     +--rw AutoBwMin?       uint32
         |     +--ro adminStatus?                 enumeration
         |     +--ro operStatus?                  enumeration
         |     +--rw tunnelInterface
         |     |  +--rw statEnable?   boolean
         |     |  +--rw igpAttr
         |     |     +--rw advertiseEnable?          boolean
         |     |     +--rw shortcutType?             enumeration
         |     |     +--rw igpMetricType?            enumeration
         |     |     +--rw relativeIgpMetricValue?   int16
         |     |     +--rw absoluteIgpMetricValue?   uint16
         |     |     +--rw advertiseHoldTime?        uint32
         |     +--rw frrAttr
         |     |  +--rw frrEnable?          boolean
         |     |  +--rw bwProtEnable?       boolean
         |     |  +--rw frrBandwidth?       uint32
         |     |  +--rw frrSetupPriority?   uint32
         |     |  +--rw frrHoldPriority?    uint32
         |     +--rw bypassAttr
         |        +--rw bypassEnable?       boolean
         |        +--rw bypassProtectIFs
         |           +--rw bypassProtectIF* [bypassProtectIFName]
         |              +--rw bypassProtectIFName    ifName
```

3.7. RSVP-TE Global Configuration

RSVP-TE global configuration includes the global parameters for the RSVP-TE signaling such as different timer parameters, hello capability, SRefresh capability, etc.

```
         +--rw rsvpTESite
         |  +--rw helloEnable?                        boolean
         |  +--rw maxHelloMissTimes?                  uint32
         |  +--rw helloInterval?                      uint32
         |  +--rw supportGREnable?                    boolean
         |  +--rw keepMultiplier?                     uint32
         |  +--rw refreshInterval?                    uint32
         |  +--rw srefreshEnable?                     boolean
         |  +--rw retransmissionInterval?             uint32
         |  +--rw retransmissionIncrementValue?       uint32
         |  +--rw challengeRetransmissionInterval?    uint32
         |  +--rw maxChallengeMissTimes?              uint32
```

3.8. RSVP-TE Interface Configuration

RSVP-TE interface configuration includes the parameter of the RSVP-TE interface such as interface name, hello capability, auto FRR mode and authentication.

```
         +--rw rsvpInterfaces
         |  +--rw rsvpInterface* [interfaceName]
         |     +--rw interfaceName     ifName
         |     +--rw helloEnable?      boolean
         |     +--ro autoFrrMode?      enumeration
         |     +--rw authentication
         |        +--rw authEnable?       boolean
         |        +--rw authMD5Key?       string
         |        +--rw authLifetime?     uint32
         |        +--rw authHandshake?    string
         |        +--rw authWindowSize?   uint32
```

3.9. CSPF Configuration

CSPF configuration includes the IGP choice, tie-breaking policy, metric type, SRLG policy for the MPLS TE path calculation.

```
      +--rw cspfCfg
      |  +--rw enableCspf?               boolean
      |  +--rw preferredIgp?             enumeration
      |  +--rw preferredOspfProcessId?   uint32
      |  +--rw preferredOspfAreaId?      boolean
      |  +--rw preferredIsisProcessId?   uint32
      |  +--rw preferredIsisLevel?       enumeration
      |  +--rw tiebreaking?              enumeration
      |  +--rw pathMetricType?           enumeration
      |  +--rw srlgPathCalcMode?         enumeration
```

3.10. P2MP TE Tunnel Template Configuration

P2MP TE tunnel template will be used to set up P2MP TE tunnel triggered by multicast service such as BGP-base MVPN defined in [RFC6514].

```
      +--rw p2mpTeTemplates
         +--rw p2mpTeTemplate* [templateName]
            +--rw templateName             string
            +--rw recordRouteMode?         enumeration
            +--rw resvStyle?               enumeration
            +--rw setupPriority?           uint8
            +--rw holdPriority?            uint8
            +--rw bandwidth?               uint32
            +--rw reoptimization?          boolean
            +--rw reoptiFrequency?         uint32
            +--rw pathMetricType?          enumeration
            +--rw tieBreaking?             enumeration
            +--rw hopLimit?                uint32
            +--ro includeAllAffinity?      string
            +--rw includeAnyAffinity?      string
            +--rw excludeAnyAffinity?      string
            +--rw leafListName             string
            +--rw mplsTeP2mpTemplateFrr
               +--rw frrEnable?          boolean
               +--rw bwProtEnable?       boolean
               +--rw frrBandwidth?       uint32
               +--rw frrSetupPriority?   uint32
               +--rw frrHoldPriority?    uint32
```

4. MPLS TE Yang Module

```
module mplste {
    namespace "urn:huawei:params:xml:ns:yang:mplste";
    // replace with IANA namespace when assigned - urn:ietf:params:xml:ns:yang:1
    prefix "mplste";
  import ietf-inet-types {
    prefix inet;
  }
    organization
      "Huawei Technologies Co., Ltd.";
    contact
      "jescia.chenxia@huawei.com
       lizhenbin@huawei.com
       zengxinzong@huawei.com";
    description
      "This YANG module defines the generic configuration
     data for MPLS TE, which is common across all of the vendor
     implementations of the protocol. It is intended that the module
     will be extended by vendors to define vendor-specific
     MPLS TE configuration parameters.";
  revision 2014-08-16 {
    description
      "Initial revision.";
  }

  typedef ifName {
        description "ifName is like ethernet1/1/1/1";
        type string {
            length "1..63";
        }
    }

    container mplsTe {

        container mplsTeSite {

            description "MPLS TE basic Configuration.";

            leaf enablePeriFl {
                description "Capability of flooding TE link bandwidth periodical
ly. When TE LSP reserves or releases bandwidth with the changing rate not reachi
ng the flooding threshold of TE link, it will not flood bandwidth to network. In
 order to flood bandwidth to network as soon as possible, and to avoid waste of
network resource caused by frequent flooding, it will flood changed bandwidth to
 network periodically.";
                config "true";
                default "false";
                type boolean;
            }
            leaf floodingInterval {
                description "Interval at which a TE interface floods bandwidth o
ver the network. The interval ranges from 10s to 43200s, and the default value i
s 30s. The bandwidth that a TE LSP reserves or releases on an interface may be o
nly a small part compared with the reservable bandwidth of the interface, which
does not reach the flooding threshold of the interface. To flood the interface b
andwidth over the network and avoid frequent flooding, the TE NE is configured t
o flood bandwidth over the network periodically.";
                config "true";
```

```
                default "30";
                type uint32 {
                    range "10..43200";
                }
            }
            leaf switchDelay {
                description "Delay time for switching the TE traffic from a Prim
ary CR-LSP to a Modified CR-LSP. The delay time ranges from 0s to 120000s, and t
he default delay value is 5000s.";
                config "true";
                default "5000";
                type uint32 {
                    range "0..120000";
                }
            }
            leaf autoFrrEnable {
                description "Specifies the enabling state of auto FRR.";
                config "true";
                default "false";
                type boolean;
            }
            leaf autobwEnable {
                description "Specifies the enabling state of auto bandwidth.";
                config "true";
                default "false";
                type boolean;
            }
            leaf autobwInterval {
                description "Set interval of Auto bandwidth sampling.";
                config "true";
                default "300";
                type uint32 {
                    range "1..604800";
                }
            }
        }

        container teLinks {

            list teLink {

                key "interfaceName";
                max-elements "unbounded";
                min-elements "0";
                description "TELINK";

                leaf interfaceName {
                    description "Specifies the name of a physical interface wher
e TE is enabled.";
                    config "true";
                    type ifName;
                }
                leaf teIfMaxreservablebandwidth {
                    description "Specifies the maximum reserved bandwidth (kbit/
s) of TE interfaces. The value ranges from 0 kbit/s to 4294967295 kbit/s. By def
ault, the value is 0 Kbit/s. The limit of interface bandwidth is as follows: The
 configured bandwidth on a TE interface cannot exceed the physical bandwidth of
the interface. Otherwise, the NE prompts an error.";
                    config "true";
                    default "0";
                    type uint32 {
                        range "0..4294967295";
                    }
                }
                leaf teIfBc0bandwidth {
                    description "Bc0 bandwidth value in kbps";
                    config "true";
                    default "0";
                    type uint32 {
                        range "0..4294967295";
                    }
                }
                leaf adminGroups {
                    description "Specifies interface management attributes. They
 are total 32 bits. Each bit is one attribute that is established on a TE LSP. S
tandby LSPs are selected based on the limit of the LSP affinity attribute.
                    It’s hexadecimal string.";
                    config "true";
                    default "00000000";
                    type string {
                        length "8";
                  }
                }
                container srlgGroups {

                    list srlgGroup {

                        key "teIfSrlgValue";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Specifies a risk-shared link group.";

                        leaf teIfSrlgValue {
                            description "Specifies a risk-shared link group numb
er.";
                            config "true";
                            type uint32 {
                                range "0..4294967295";
                            }
                        }
                    }

                }

                leaf bwChangeThresholdDown {
                    description "Specifies the flooding threshold of bandwidth c
onsumption percentage. The value ranges from 0?? to 100??. By default, the value
 is 10??. Establishing TE LSPs consumes the bandwidth of the TE interface, and t
he available bandwidth of the interface is therefore reduced. If the percentage
of the reduced bandwidth is larger than the configured flooding threshold, the b
andwidth of the TE interface will be flooded to the network.";
```

```
                    config "true";
                    default "10";
                    type uint32 {
                        range "0..100";
                    }
                }
                leaf bwChangeThresholdUp {
                    description "Specifies the flooding threshold of the percent
age of released bandwidth compared with the available interface bandwidth. The v
alue ranges from 0?? to 100??. By default, the value is 10??. When a TE LSP of a
n interface is removed, the bandwidth of the TE LSP is released. The available b
andwidth of the interface is therefore increased. If the percentage of the incre
ased bandwidth is larger than the configured flooding threshold, the bandwidth o
f the TE interface will be flooded over the network.";
                    config "true";
                    default "10";
                    type uint32 {
                        range "0..100";
                    }
                }
                leaf teIfMetric {
                    description "Specifies the Value of TE interface metric. The
 value ranges from 1 to 16777215. By default, the value is 1. The link with smal
lest metric value is used preferentially as a TE LSP.";
                    config "true";
                    type uint32 {
                        range "1..16777215";
                    }
                }
                leaf autoFrrMode {
                    description "Auto FRR protection modes: global inheritance,
link protection, node protection and self-adapting. By default, global AutoFRR i
s global inheritance. Link protection: A bypass tunnel can be used to protect on
ly the links where the outbound interface of the primary LSP resides. Node prote
ction: A bypass tunnel can be used to protect the downstream node of the links w
here the outbound interface of the primary LSP resides.Self-adapting??A bypass t
unnel can be dynamically selected node protection or link protection according t
o the network conditions.";
                    config "true";
                    default "DEFAULT";
                    type enumeration {
                        enum DEFAULT {
                            value "0";
                            description "DEFAULT:";
                        }
                        enum LINK {
                            value "1";
                            description "LINK:";
                        }
                        enum NODE {
                            value "2";
                            description "NODE:";
                        }
                        enum DISABLE {
                            value "3";
                            description "DISABLE:";
                        }
                    }
                }
            }
        }

        container explicitPaths {
```

```
            list explicitPath {

                key "explicitPathName";
                max-elements "unbounded";
                min-elements "0";
                description "Explicit path.";

                leaf explicitPathName {
                    description "Name of an explicit path.";
                    config "true";
                    type string {
                        length "1..31";
                        pattern "^[^ \?]*$";
                    }
                }
                container explicitPathHops {

                    list explicitPathHop {

                        key "mplsTunnelHopIndex";
                        max-elements "unbounded";
                        min-elements "0";

                        leaf mplsTunnelHopIndex {
                            description "Hop index of an explicit path.";
                            config "true";
                            type uint32 {
                                range "1..65535";
                            }
                        }
                        leaf mplsTunnelHopIpAddr {
                            description "IP address of hop.";
                            config "true";
                            mandatory "true";
                            type inet:ipv4-address;
                        }
                        leaf mplsTunnelHopType {
                            description "Specifies an LSP route selection types
based on the local hop. Strict type: Only an LSP route that includes the local h
op can be selected. Loose type: An LSP route that includes the local node is sel
ected preferentially. If the local hop does not meet path limits, it will be not
 included in the selected route.  Excluding type: Only an LSP route that does no
t include the local hop can be selected.";
                            config "true";
                            default "includeStrict";
                            type enumeration {
                                enum includeStrict {
                                    value "0";
                                    description "Strictly included.";
                                }
                                enum includeLoose {
                                    value "1";
                                    description "Loosely included.";
                                }
                                enum exclude {
                                    value "2";
                                    description "Excluded.";
                                }
                            }
                        }
                        leaf mplsTunnelHopAddrType {
                            description "Address type.";
                            config "true";
                            default "IPV4";
                            type enumeration {
                                enum IPV4 {
                                    value "0";
                                    description "IPV4:";
                                }
                            }
                        }
                    }

                }

            }

        }

        container p2mpLeafLists {

            list p2mpLeafList {

                key "leafListName";
                max-elements "unbounded";
                min-elements "0";
                description "Leaf list.";

                leaf leafListName {
                    description "Name of leaf list.";
                    config "true";
                    type string {
                        length "1..31";
                        pattern "^[^ \?]*$";
                    }
                }
                container leafs {

                    list leaf {

                        key "leafIpAddr";
                        max-elements "unbounded";
                        min-elements "0";
            description "Leaf.";

                        leaf leafIpAddr {
                            description "Leaf Address.";
                            config "true";
                            type inet:ipv4-address;
                        }
                        leaf explicitPathName {
                            description "Explicit path name used for a leaf.";
                            config "true";
                            type string {
                                length "0..31";
                            }
                        }
                    }

                }

            }

        }

        container rsvpTeTunnels {

            list rsvpTeTunnel {

                key "tunnelName";
                max-elements "unbounded";
                min-elements "0";
                description "TE E2E Dynamic unicast tunnel.";

                leaf tunnelName {
                    description "Tunnel name. A tunnel name is unique among all
tunnels established on a node. Specifies a tunnel name whose prefix is Tunnel. T
unnel interfaces can be named in the following formats: 1. Tunnel x/y/z ('x' ran
ges from 0 to 16, 'y' ranges from 0 to 15, and 'z' ranges from 0 to 65535); 2.Tu
nnelN ('N' ranges from 0 to 4294967295).Under the environment of clusters, Tunne
l interfaces can be named in the following formats: 1. Tunnelx/y/z/k ('x' ranges
 from 0 to 16, 'y' ranges from 0 to 16,'z' ranges from 0 to 15, and 'k' ranges f
rom 0 to 65535); 2.TunnelN ('N' ranges from 0 to 4294967295).";
                    config "true";
                    type string {
                        length "0..63";
                        pattern "^[^ ]*$";
                    }
                }
                leaf mplsTunnelIngressLSRId {
                    description "Specifies ingress LSR ID of the tunnel.";
                    config "false";
                    type inet:ipv4-address;
                }
                leaf mplsTunnelEgressLSRId {
                    description "Specifies egress LSR ID of the tunnel.";
                    config "true";
                    type inet:ipv4-address;
                }
                leaf mplsTunnelIndex {
                    description "Session ID of a tunnel.";
                    config "true";
                    type uint16;
                }
                leaf mplsTunnelBandwidth {
                    description "Specifies a tunnel bandwidth. The value ranges
from 0 kbit/s to 4000000000 kbit/s. By default, the value is 0 kbit/s.";
                    config "true";
                    default "0";
                    type uint32 {
                        range "0..4000000000";
                    }
                }
                leaf mplsTeTunnelSetupPriority {
                    description "Specifies a tunnel setup priority. The value ra
nges from 0 to 7. By default, the value is 7. The smaller the value, the higher
the setup priority. 0 is the highest priority. Limit: the setup priority of a tu
nnel must be equal to or smaller than its holding priority.";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf holdPriority {
                    description "Specifies a tunnel holding priority. The value
ranges from 0 to 7. By default, the value is 7. The smaller the value, the highe
r the holding priority. 0 is the highest priority. Limit: The holding priority o
f a tunnel must be equal to or larger than its setup priority.";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf hotStandbyEnable {
                    description "Enabling of hot standby for protecting TE tunne
ls. When an active LSP is set up successfully, a standby LSP that meets certain
limits will be set up to protect the active LSP. When the active LSP fails, the
traffic on the active LSP will be switched to the standby LSP.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf hsbRevertiveMode {
                    description "hot-standby revertive??There are two revert mod
es,e.g. revertive and non-revertive.The default mode is revertive.";
                    config "true";
                    default "revertive";
                    type enumeration {
                        enum revertive {
                            value "0";
                            description "revertive";
                        }
                        enum nonRevertive {
```

```
                            value "1";
                            description "non-evertive";
                        }
                    }
                }
                leaf hotStandbyWtr {
                    description "Time of waiting recovering back to primary LSP.
 Its range is 0˜2592000, default is 10. When hot-standby backup is in use, after
 primary LSP restores, the traffic will switch to primary LSP after waiting some
 time instead of switching to primary LSP immediately. This is to avoid frequent
 switching between primary LSP and backup LSP caused by network flapping.";
                    config "true";
                    default "10";
                    type uint32 {
                        range "0..2592000";
                    }
                }
                leaf ordinaryEnable {
                    description "Specifies a tunnel ordinary backup protection c
apability. When it is enabled, and the primary LSP fails, a backup LSP that meet
s certain limits will be set up. Then, the traffic on the primary LSP will be sw
itched to the backup LSP. ";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf bestEffortEnable {
                    description "Best-effort path protection of tunnels. When be
st-effort path is enabled for a TE tunnel, and both active and standby LSP fail,
 an LSP will be set up in the best effort method.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf disableCspf {
                    description "disable Cspf of a tunnel";
                    config "true";
                    default "false";
                    type boolean;
                }
                container tunnelPaths {

                    list tunnelPath {

                        key "pathType";
                        max-elements "unbounded";
                        min-elements "0";
                        description "Path configuration of a tunnel.";

                        leaf pathType {
                            description "Path role of a tunnel. The available op
tions are primary(used by primary LSP), hot-standby(used by hot-standby backup L
SP), ordinary(used by ordinary backup LSP), and best-effort(used by best-effort
LSP).";
                            config "true";
                            type enumeration {
                                enum primary {
                                    value "0";
                                    description "Primary path.";
                                }
```

```
                                enum hot_standby {
                                    value "1";
                                    description "Hot standby path.";
                                }
                                enum ordinary {
                                    value "2";
                                    description "Ordinary backup path.";
                                }
                                enum best_effort {
                                    value "3";
                                    description "Best-effort path.";
                                }
                            }
                        }
                        leaf explicitPathName {
                            description "Name of an explicit path.";
                            config "true";
                            type string {
                                length "0..31";
                            }
                        }
                        leaf includeAll {
                            description "Administrative group attribute of an LS
P (includeAll).It’s hexadecimal string.";
                            config "false";
                            default "00000000";
                            type string {
                                length "8";
                            }
                        }
                        leaf includeAny {
                            description "Administrative group attribute of an LS
P (includeAny).It’s hexadecimal string.";
                            config "true";
                            default "00000000";
                            type string {
                                length "8";
                            }
                        }
                        leaf excludeAny {
                            description "Tunnel path management attribute: Exclu
de-any. The value of this attribute ranges from 0x0 to 0xFFFFFFFF. By default, t
he value is 0x0. The management attribute is a 32-bit vector. If the management
attribute of a link contains any bit in the Exclude-any attribute field of an LS
P, the link cannot be a candidate LSP link.It’s hexadecimal string.";
                            config "true";
                            default "00000000";
                            type string {
                                length "8";
                            }
                        }
                        leaf hopLimit {
                            description "Number limit of hops in a tunnel path.
The value ranges from 1 to 32. By default, the value is 32.";
                            config "true";
                            default "32";
                            type uint32 {
                                range "1..32";
                            }
                        }
                    }

                }

                leaf resvStyle {
                    description "Tunnel reservation styles. SE style: shared exp
licit style; FF: fixed filter style. The default tunnel reservation style is SE.
";
                    config "true";
                    default "SE";
                    type enumeration {
                        enum SE {
                            value "0";
                            description "Shared Explicit Style:";
                        }
                        enum FF {
                            value "1";
                            description "Fixed Filter Style:";
                        }
                    }
                }
                leaf mplsTunnelRecordRoute {
                    description "Route record mode of a tunnel. No-route record
mode,  route recording mode, and route and label record mode.";
                    config "true";
                    default "DISABLE";
                    type enumeration {
                        enum DISABLE {
                            value "0";
                            description "DISABLE:";
                        }
                        enum RECORD_ROUTE_ONLY {
                            value "1";
                            description "The LSP records routes only.";
                        }
                        enum RECORD_LABEL {
                            value "2";
                            description "The LSP records both routes and labels.
";
                        }
                    }
                }
                leaf reoptimization {
                    description "Auto reoptimization enabling state of TE Tunnel
.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf reoptiFrequency {
                    description "Frequency of auto reoptimization, its range is
60-604800.";
                    config "true";
                    default "3600";
                    type uint32 {
                        range "60..604800";
                    }
                }
                leaf tieBreaking {
                    description "Routing rules for a tunnel with multiple equal-
cost routes. Random: Select a link randomly. least fill: Select the link with sm
allest bandwidth usage. most fill: Select the link with biggest bandwidth usage.
 By default, routing rules are inherited from the global MPLS TE routing rules.
If multiple paths meet certain limits, a path will be selected based on the prec
eding rules.";
                    config "true";
                    default "DEFAULT";
                    type enumeration {
                        enum LEASTFILL {
                            value "0";
                            description "LEASTFILL:The link with the smallest ba
ndwidth occupation ratio is selected in the case of equal conditions.";
                        }
                        enum MOSTFILL {
                            value "1";
                            description "MOSTFILL:The link with the largest band
width occupation ratio is selected in the case of equal conditions.";
                        }
                        enum RANDOM {
                            value "2";
                            description "RANDOM:Links are selected randomly.";
                        }
                        enum DEFAULT {
                            value "3";
                            description "Inherit from global configuration.";
                        }
                    }
                }
                leaf pathMetricType {
                    description "Referenced metric type of one link for calculat
ing path when creating TE tunnels. The available options are DEFAULT, IGP and TE
, default is inheriting from global configuration.";
                    config "true";
                    default "NONE";
                    type enumeration {
                        enum NONE {
                            value "0";
                            description "Inherit from global configuration.";
                        }
                        enum IGP {
                            value "1";
                            description "IGP";
                        }
                        enum TE {
                            value "2";
                            description "TE";
                        }
                    }
                }
                container AutoBandwidths {

                    container AutoBandwidth {

                        description "Auto bandwidth.";

                        leaf AutoBwMode {
                            description "Auto bandwidth mode.";
                            config "true";
                            default "DISABLE";
                            type enumeration {
                                enum ADJUSTMENT {
                                    value "0";
                                    description "Adjustment mode??a lsp will be
created??if all of the conditions are satisfied.";
                                }
                                enum COLLECTBW {
                                    value "1";
                                    description "Collect mode??only collect band
width in this mode.";
                                }
                                enum DISABLE {
                                    value "2";
                                    description "Disable capability of Auto band
width.";
                                }
                            }
                        }
                        leaf thresholdPerc {
                            description "Set the threshold of the adjustment ban
dwidth.";

                            config "true";
                            default "0";
                            type uint32 {
                                range "0..100";
                            }
                        }
                        leaf AutoBwFreq {
                            description "Set Frequency of Auto bandwidth.";
                            config "true";
                            default "86400";
                            type uint32 {
                                range "300..604800";
                            }
                        }
                        leaf AutoBwMax {
                            description "Max Bandwidth.";
                            config "true";
                            default "4000000000";
                            type uint32 {
                                range "0..4000000000";
                            }
                        }
                        leaf AutoBwMin {
                            description "Min Bandwidth.";
                            config "true";
                            default "0";
                            type uint32 {
                                range "0..4000000000";
                            }
                        }
                    }

                }
                leaf adminStatus {
                    description "Administrative state of a tunnel--(UP??Down)";
                    config "false";
                    default "up";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
                        }
                    }
                }
                leaf operStatus {
                    description "Operation status of a tunnel--(UP??Down)";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
                        }
                    }
                }
                container tunnelInterface {

                    description "TE tunnel interface.";
                    leaf statEnable {
                        description "Traffic statistic enabling state.";
                        config "true";
                        type boolean;
                    }
                    container igpAttr {

                        description "IGP attribute of tunnel interface.";

                        leaf advertiseEnable {
                            description "Tunnel interface forwarding adjacency b
ased on tunnel interfaces. An MPLE TE tunnel can be advertised as a virtual link
 over an IGP network. The virtual link can participate in route calculation.";
                            config "true";
                            default "false";
                            type boolean;
                        }
                        leaf shortcutType {
                            description "Tunnel interface shortcut types. Disabl
ed: This function is not enabled, and virtual TE tunnel interfaces do not partic
ipate in route calculation. OSPF type: Virtual TE tunnel interfaces participate
in OSPF route calculation. ISIS type: Virtual TE tunnel interfaces participates
in ISIS route calculation. Limit: IGP Shortcut and forwarding adjacency cannot f
unction together.";
                            config "true";
                            default "disable";
                            type enumeration {
                                enum disable {
                                    value "0";
                                    description "disable:";
                                }
                                enum ospf {
                                    value "1";
                                    description "ospf:";
                                }
                                enum isis {
                                    value "2";
                                    description "isis:";
                                }
                                enum both {
                                    value "3";
                                    description "both:";
                                }
                            }
                        }
                        leaf igpMetricType {
                            description "IGP metric types of tunnel interfaces.
Relative metric type and absolute metric type.";
                            config "true";
                            default "relative";
                            type enumeration {
                                enum absolute {
                                    value "0";
                                    description "absolute:";
                                }
                                enum relative {
                                    value "1";
                                    description "relative:";
                                }
                            }
                        }
                        leaf relativeIgpMetricValue {
                            description "IGP relative metric value of a tunnel i
nterface.";
                            config "true";
                            default "0";
                            type int16 {
                                range "-10..10";
                            }
                        }
                        leaf absoluteIgpMetricValue {
                            description "IGP absolute metric value of a tunnel i
nterface.";
                            config "true";
                            default "1";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf advertiseHoldTime {
                            description "When Tunnel get down, the Time of Notif
ying tunnel as virtual link to other routers.";
                            config "true";
                            default "0";
                            type uint32 {
                                range "0..4294967295";
                            }
                        }
                    }

                }

                container frrAttr {

                    description "Fast reroute attribute.";

                    leaf frrEnable {
                        description "Request of fast reroute capability.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf bwProtEnable {
                        description "The tunnel with fast reroute capability req
uests bandwidth protection.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf frrBandwidth {
                        description "FRR-protection bandwidth  (kbits/s) request
ed by an active tunnel. The value ranges from 0 kbit/s to 4000000000 kbit/s. by
default, the value is 0 Kbit/s. The value cannot exceed the bandwidth of the act
ive tunnel.";
                        config "true";
                        default "0";
                        type uint32 {
                            range "0..4000000000";
                        }
                    }
                    leaf frrSetupPriority {
                        description "Setup priority of FRR-protection tunnels. T
he value ranges from 0 to 7. By default, the value is 7. The smaller the value,
the higher the setup priority. 0 is the highest priority. Limit: The protection
tunnel setup priority cannot exceed the setup priority of the active tunnel.";
                        config "true";
                        default "7";
                        type uint32 {
                            range "0..7";
                        }
                    }
                    leaf frrHoldPriority {
                        description "Holding priority of FRR protection tunnels.
 The value ranges from 0 to 7. The smaller the value, the higher the priority. T
he value 0 is the highest priority. Limit: The protection tunnel holding priorit
y cannot exceed the active tunnel holding priority.";
                        config "true";
                        default "7";
                        type uint32 {
                            range "0..7";
                        }
                    }
                }

                container bypassAttr {

                    description "Bypss tunnel attribute.";

                    leaf bypassEnable {
                        description "Bypass tunnel enabling or disabling. A bypa
ss tunnel can be enabled to protect a tunnel that requests FRR protection. Limit
: A bypass tunnel cannot request FRR protection.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    container bypassProtectIFs {

                        list bypassProtectIF {

                            key "bypassProtectIFName";
                            max-elements "unbounded";
                            min-elements "0";
                            description "Specifies a list of interfaces that can
 be protected by a bypass tunnel.";

                            leaf bypassProtectIFName {
                                description "Specifies the name of an interface
that can be protected by a tunnel enabled with the bypass function.";
```

```
                                config "true";
                                type ifName;
                            }
                        }

                    }

                }

            }

        }

        container rsvpTESite {

            description "RSVP configuration and status.";

            leaf helloEnable {
                description "Hello enabling state.";
                config "true";
                default "false";
                type boolean;
            }
            leaf maxHelloMissTimes {
                description "Maximum number of Hello packet loss times. The valu
e ranges from 3 to 10, and the default value is 3. When the number of Hello pack
et refreshing (from the NE) times exceeds the configured maximum number, and the
 NE does not receive a response from its peer end, the NE considers that the ses
sion with the peer end is lost.";
                config "true";
                default "3";
                type uint32 {
                    range "3..10";
                }
            }
            leaf helloInterval {
                description "Interval for refreshing Hello packets. It is the in
terval of refreshing a Hello packet from an NE to its peer after a Hello session
 is established. The value ranges from 1s to 25s. By default, the value is 3s.";
                config "true";
                default "3";
                type uint32 {
                    range "1..25";
                }
            }
            leaf supportGREnable {
                description "Support GR enabling state.";
                config "true";
                default "false";
                type boolean;
            }
            leaf keepMultiplier {
                description "Timeout multiplier of soft state PSB or RSB. The va
lue ranges from 3 to 255. By default, the value is 3. If the local NE does not r
eceive a packet for refreshing the TE LSP soft state from its upstream or downst
ream neighbor in the period (that is a specified multiplier of the refreshing cy
cle), the NE considers that the soft state times out and will delete the TE LSP.
";
                config "true";
```

```
                default "3";
                type uint32 {
                    range "3..255";
                }
            }
            leaf refreshInterval {
                description "Cycle of refreshing a PATH or a RESV message (for s
oft state maintenance). The value ranges from 10 to 65535. By default, the value
 is 30.";
                config "true";
                default "30";
                type uint32 {
                    range "10..65535";
                }
            }
            leaf srefreshEnable {
                description "Summary refresh enabling state.";
                config "true";
                default "false";
                type boolean;
            }
            leaf retransmissionInterval {
                description "Interval at which interface summary messages are re
transmitted. The value ranges from 500 ms to 5000 ms. By default, the value is 5
000 ms.";
                config "true";
                default "5000";
                type uint32 {
                    range "500..5000";
                }
            }
            leaf retransmissionIncrementValue {
                description "Incremental value for retransmitting interface summ
ary. The value ranges from 1 to 10. By default, the value is 1. If an NE does no
t receive a response from the peer end after sending a message to the peer end,
the NE will start the retransmission mechanism. The retransmission interval incr
eases with the increase of the retransmission incremental value. The formula is
as following: retransmission interval = last retransmission interval * (1 + retr
ansmission incremental value).";
                config "true";
                default "1";
                type uint32 {
                    range "1..10";
                }
            }
            leaf challengeRetransmissionInterval {
                description "Challenge message retransmission interval. The valu
e ranges from  500 ms to 10000 ms. By default, the value is 1000 ms. If the NE d
oes not receive a response message from the peer end after sending a Challenge m
essage for handshake authentication to the peer end, the NE will retransmit the
Challenge message.";
                config "true";
                default "1000";
                type uint32 {
                    range "500..10000";
                }
            }
            leaf maxChallengeMissTimes {
                description "Maximum challenge loss times. The value ranges from
 1 to 10. By default, the value is 3. If the times of a retransmitting a Challen
ge message from the NE to its neighbor exceed the maximum challenge loss times,
the NE considers that the handshake cannot be established and will not send a Ch
allenge message again.";
```

```
                config "true";
                default "3";
                type uint32 {
                    range "1..10";
                }
            }
        }

        container rsvpInterfaces {

            list rsvpInterface {

                key "interfaceName";
                max-elements "unbounded";
                min-elements "0";
                description "RSVP interface configuration.";

                leaf interfaceName {
                    description "Interface name.";
                    config "true";
                    type ifName;
                }
                leaf helloEnable {
                    description "Hello enabling state.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf autoFrrMode {
                    description "Auto FRR protection modes: global inheritance,
link protection, node protection and self-adapting. By default, global AutoFRR i
s global inheritance. Link protection: A bypass tunnel can be used to protect on
ly the links where the outbound interface of the primary LSP resides. Node prote
ction: A bypass tunnel can be used to protect the downstream node of the links w
here the outbound interface of the primary LSP resides.Self-adapting??A bypass t
unnel can be dynamically selected node protection or link protection according t
o the network conditions.";
                    config "false";
                    default "DEFAULT";
                    type enumeration {
                        enum DEFAULT {
                            value "0";
                            description "DEFAULT:";
                        }
                        enum LINK {
                            value "1";
                            description "LINK:";
                        }
                        enum NODE {
                            value "2";
                            description "NODE:";
                        }
                        enum DISABLE {
                            value "3";
                            description "DISABLE:";
                        }
                    }
                }
                container authentication {

                    description "Neighbor configuration of RSVP TE authenticatio
n.";

                    leaf authEnable {
                        description "Specifies the enabling state of RSVP TE int
erface authentication.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf authMD5Key {
                        description "Authentication key of a RSVP TE interface.
The key length ranges from 1 to 255 characters. When the key is configured, the
RSVP packets that the interface sends out carry the authentication information t
hat are calculated based on the authentication key by using the MD5 algorithm. T
he packets the interface receives are also verified based on the authentication
key. The two ends of a TE link must be configured with the same authentication k
ey, otherwise, RSVP packets cannot pass through the link.";
                        config "true";
                        type string {
                    length "1..255";
                            pattern "^[^ ]+$";
                        }
                    }
                    leaf authLifetime {
                        description "Authentication lifetime. The value ranges f
rom 1 to 86399, in seconds. By default, the value is 1800s. During authenticatio
n lifetime, RSVP authentication lifetime will be reset after a RSVP packet is re
ceived. If no RSVP packet is received when the authentication lifetime times out
, the RSVP neighbor will delete the authentication relationship to prevent persi
stent authentication.";
                        config "true";
                        default "1800";
                        type uint32 {
                            range "1..86399";
                        }
                    }
                    leaf authHandshake {
                        description "Handshake authentication enable. When hands
hake enabled, the handshake mechanism is configured. When the authentication seq
uence number of an RSVP packet is out of order, a handshake will be initiated an
d an authentication will be re-negotiated.";
                        config "true";
                        type string {
                            length "8..40";
                            pattern "^[^ ]+$";
                        }
                    }
                    leaf authWindowSize {
                        description "Size of an authentication window that is se
t to prevent information loss caused by packet disorder. The value of the size r
anges from 1 to 64. By default, the value is 1. If the packet sequence number ex
ceeds the window size, the packet is considered to be out of order.";
                        config "true";
                        default "1";
                        type uint32 {
                            range "1..64";
                        }
                    }
                }
```

```
            }
        }
        container cspfCfg {

            description "CSPF configuration.";

            leaf enableCspf {
                description "The enabling state of CSPF capability.";
                config "true";
                default "false";
                type boolean;
            }
            leaf preferredIgp {
                description "Configurable entry: preferred Igp";
                config "true";
                default "ospf";
                type enumeration {
                    enum ospf {
                        value "0";
                        description "ospf:";
                    }
                    enum isis {
                        value "1";
                        description "isis:";
                    }
                }
            }
            leaf preferredOspfProcessId {
                description "preferred ospf process id";
                config "true";
                type uint32 {
                    range "1..4294967295";
                }
            }
            leaf preferredOspfAreaId {
                description "preferred ospf area id";
                config "true";
                type boolean;
            }
            leaf preferredIsisProcessId {
                description "preferred isis process id";
                config "true";
                type uint32 {
                    range "1..4294967295";
                }
            }
            leaf preferredIsisLevel {
                description "preferred isis level";
                config "true";
                type enumeration {
                    enum invalid {
                        value "0";
                        description "ISIS invalid";
                    }
                    enum level1 {
                        value "1";
                        description "ISIS Level-1";
                    }
                    enum level2 {
                        value "2";
                        description "ISIS Level-2";
                    }
                }
            }
            leaf tiebreaking {
                description "Rule of selecting multiple equivalent routes.";
                config "true";
                default "RANDOM";
                type enumeration {
                    enum LEASTFILL {
                        value "0";
                        description "LEASTFILL:The link with the smallest bandwi
dth occupation ratio is selected in the case of equal conditions.";
                    }
                    enum MOSTFILL {
                        value "1";
                        description "MOSTFILL:The link with the largest bandwidt
h occupation ratio is selected in the case of equal conditions.";
                    }
                    enum RANDOM {
                        value "2";
                        description "RANDOM:Links are selected randomly.";
                    }
                }
            }
            leaf pathMetricType {
                description "pre-calc path metric type, TE or IGP.";
                config "true";
                default "TE";
                type enumeration {
                    enum IGP {
                        value "0";
                        description "IGP";
                    }
                    enum TE {
                        value "1";
                        description "TE";
                    }
                }
            }
            leaf srlgPathCalcMode {
                description "cspf srlg path calculating mode";
                config "true";
                default "default";
                type enumeration {
                    enum default {
                        value "0";
                        description "none: Don’t care SRLG when calculte path.";
                    }
                    enum preferred {
                        value "1";
                        description "preferred: Try to exclude link in the same
SRLG with excluded path.";
                    }
                    enum strict {
                        value "2";
                        description "strict: Never use link in same SRLG with ex
clude path.";
                    }
                }
            }
        }

        container p2mpTeTemplates {

            list p2mpTeTemplate {

                key "templateName";
                max-elements "unbounded";
                min-elements "0";
                description "P2MP TE configuration Template.";

                leaf templateName {
                    description "Name of p2mp te configuration template.";
                    config "true";
                    type string {
                        length "0..31";
                        pattern "^[^ ]*$";
                    }
                }
                leaf recordRouteMode {
                    description "Route record mode. The available options are no
t record, recording routes only, and recording routes and labels.";
                    config "true";
                    default "DISABLE";
                    type enumeration {
                        enum DISABLE {
                            value "0";
                            description "DISABLE:";
                        }
                        enum RECORD_ROUTE_ONLY {
                            value "1";
                            description "The LSP records routes only.";
                        }
                        enum RECORD_LABEL {
                            value "2";
                            description "The LSP records both routes and labels.
";
                        }
                    }
                }
                leaf resvStyle {
                    description "Tunnel reservation styles. SE style: shared exp
licit style; FF: fixed filter style. The default tunnel reservation style is SE.
";
                    config "true";
                    default "SE";
                    type enumeration {
                        enum SE {
                            value "0";
                            description "Shared Explicit Style:";
                        }
                        enum FF {
                            value "1";
                            description "Fixed Filter Style:";
                        }
                    }
                }
                leaf setupPriority {
                    description "Specifies a tunnel setup priority. The value ra
nges from 0 to 7. By default, the value is 7. The smaller the value, the higher
the setup priority. 0 is the highest priority. Limit: the setup priority of a tu
nnel must be equal to or smaller than its holding priority.";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf holdPriority {
                    description "Specifies a tunnel holding priority. The value
ranges from 0 to 7. By default, the value is 7. The smaller the value, the highe
r the holding priority. 0 is the highest priority. Limit: The holding priority o
f a tunnel must be equal to or larger than its setup priority.";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf bandwidth {
                    description "Specifies a tunnel bandwidth. The value ranges
from 0 kbit/s to 4000000000 kbit/s. By default, the value is 0 kbit/s.";
                    config "true";
                    default "0";
                    type uint32 {
                        range "0..4000000000";
                    }
                }
                leaf reoptimization {
                    description "Auto reoptimization enabling state of P2MP Temp
late.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf reoptiFrequency {
                    description "Frequency of auto reoptimization, its range is
60-604800.";
                    config "true";
                    default "3600";
                    type uint32 {
                        range "60..604800";
                    }
                }
                leaf pathMetricType {
                    description "Referenced metric type of one link for calculat
ing path when creating TE tunnels. The available options are DEFAULT, IGP and TE
, default is inheriting from global configuration.";
                    config "true";
                    default "NONE";
                    type enumeration {
                        enum NONE {
                            value "0";
                            description "Inherit from global configuration.";
                        }
                        enum IGP {
                            value "1";
                            description "IGP";
                        }
                        enum TE {
                            value "2";
                            description "TE";
                        }
                    }
                }
                leaf tieBreaking {
                    description "Routing rules for a p2mp template with multiple
 equal-cost routes. Random: Select a link randomly. least fill: Select the link
with smallest bandwidth usage. most fill: Select the link with biggest bandwidth
 usage. By default, routing rules are inherited from the global MPLS TE routing
rules. If multiple paths meet certain limits, a path will be selected based on t
he preceding rules.";
                    config "true";
                    default "DEFAULT";
                    type enumeration {
                        enum LEASTFILL {
                            value "0";
                            description "LEASTFILL:The link with the smallest ba
ndwidth occupation ratio is selected in the case of equal conditions.";
                        }
                        enum MOSTFILL {
                            value "1";
                            description "MOSTFILL:The link with the largest band
width occupation ratio is selected in the case of equal conditions.";
                        }
                        enum RANDOM {
                            value "2";
```

```
                            description "RANDOM:Links are selected randomly.";
                        }
                        enum DEFAULT {
                            value "3";
                            description "Inherit from global configuration.";
                        }
                    }
                }
                leaf hopLimit {
                    description "Number limit of hops in a tunnel path. The valu
e ranges from 1 to 32. By default, the value is 32.";
                    config "true";
                    default "32";
                    type uint32 {
                        range "1..32";
                    }
                }
                leaf includeAllAffinity {
                    description "Administrative group attribute of an LSP (inclu
deAll).It’s hexadecimal string.";
                    config "false";
                    default "00000000";
                    type string {
                        length "8";
                    }
                }
                leaf includeAnyAffinity {
                    description "Administrative group attribute of an LSP (inclu
deAny).It’s hexadecimal string.";
                    config "true";
                    default "00000000";
                    type string {
                        length "8";
                    }
                }
                leaf excludeAnyAffinity {
                    description "Administrative group attribute of an LSP (exclu
deAny).It’s hexadecimal string.";
                    config "true";
                    default "00000000";
                    type string {
                        length "8";
                    }
                }
                leaf leafListName {
                    description "Specify the leaf-list.";
                    config "true";
                    mandatory "true";
                    type string {
                        length "0..31";
                    }
                }
                container mplsTeP2mpTemplateFrr {

                    description "Fast reroute attribute.";

                    leaf frrEnable {
                        description "Request of fast reroute capability.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf bwProtEnable {
                        description "The tunnel with fast reroute capability req
uests bandwidth protection.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf frrBandwidth {
                        description "FRR-protection bandwidth  (kbits/s) request
ed by an active tunnel. The value ranges from 0 kbit/s to 4000000000 kbit/s. by
default, the value is 0 Kbit/s. The value cannot exceed the bandwidth of the act
ive tunnel.";
                        config "true";
                        default "0";
                        type uint32 {
                            range "0..4000000000";
                        }
                    }
                    leaf frrSetupPriority {
                        description "Setup priority of FRR-protection tunnels. T
he value ranges from 0 to 7. By default, the value is 7. The smaller the value,
the higher the setup priority. 0 is the highest priority. Limit: The protection
tunnel setup priority cannot exceed the setup priority of the active tunnel.";
                        config "true";
                        default "7";
                        type uint32 {
                            range "0..7";
                        }
                    }
                    leaf frrHoldPriority {
                        description "Holding priority of FRR protection tunnels.
 The value ranges from 0 to 7. The smaller the value, the higher the priority. T
he value 0 is the highest priority. Limit: The protection tunnel holding priorit
y cannot exceed the active tunnel holding priority.";
                        config "true";
                        default "7";
                        type uint32 {
                            range "0..7";
                        }
                    }
                }

            }

        }
    }
}
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC4875] Aggarwal, R., Papadimitriou, D., and S. Yasukawa, "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, May 2007.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors’ Addresses

Xia Chen
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: jescia.chenxia@huawei.com

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

Email: lizhenbin@huawei.com

Xinzong Zeng
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zengxinzong@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

Z. Cui
R. Winter
NEC
H. Shah
Ciena
S. Aldrin
Huawei Technologies
M. Daikoku
KDDI

October 27, 2014

Use Cases and Requirements for MPLS-TP multi-failure protection
draft-cui-mpls-tp-mfp-use-case-and-requirements-03

Abstract

The basic survivability technique has been defined in Multiprotocol Label Switching Transport Profile (MPLS-TP) network [RFC6378]. That protocol however is limited to 1+1 and 1:1 protection, not designed to handle multi-failure protection.

This document introduces some use cases and requirements for multi-failure protection functionality.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Today's packet optical transport networks are able to concentrate large volumes of traffic onto a relatively small number of nodes and links. As a result, the failure of a single network element can potentially interrupt a large amount of traffic. For this reason, ensuring survivability through network design is an important network design objective.

The basic survivability technique has been defined in MPLS-TP network [RFC6378]. That protocol however is limited to 1+1 and 1:1 protection, not designed to handle multi-failure protection.

The multi-failure protection is required for disaster recovery, e.g., even during natural disasters and other catastrophic events such as earthquake or tsunami, the network availability must be provided especially for high-priority services such as emergency telephone calls. Existing 1+1 or 1:n protection however is limited to cover single failure and no sufficient to maintain disaster recovery.

The multi-failure protection is also required for hazardous contion, e.g., when a working path or protection path was closed by network operator for construction work, the network service will become a hazardous condition. During this condition time, if another failure (e.g. a human-error or network entities failure) is occurred on the protection path, than the operator can’t meet service level agreements (SLA). Thus, the multi-failure condition could put pressure on network operations.

On the other hand, many network operators have a very limited budget for improving network survivability. This requires a design approach, which takes budget limitations into consideration.

To increase the service availability and to reduce the backup network costs, we propose extend the 1+1 and 1:1 protection protocol to support the m:1 and m:n architecture type.

1.1. Document scope

This document describes the use cases and requirements for multi-failure protection in MPLS-TP networks without the use of control plane protocols. Existing solutions based on control plane such as GMPLS may be able to restore user traffic when multiple failures occur. Some networks however do not use full control plane operation for reasons such as service provider preferences, certain limitations or the requirement for fast service restoration (faster than achievable with control plane mechanisms). These networks are the focus of this document which defines a set of requirements for multi-failure protection not based on control plane support.

1.2. Requirements notation

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

2. m:n protection architecture

The following Figure 1 shows a protection domain with n working paths and m protection paths between ingress node LER-A and egress node LER-Z.

At the ingress node LER-A, the normal traffic is either permanently connected to its working path and may be connected to one of the protection paths (case of broadcast bridge), or is connected to either its working path or one of the protection paths (case of selector bridge). At the egress node LER-Z, the normal traffic is selected from either its working or one of the protection paths.

```
                +-----+                             +-----+
                |LER-A|     Working Path #1         |LER-Z|
                |     |=============================|     |
                |     |            ....             |     |
                |     |     Working Path #n         |     |
                |     |=============================|     |
                |     |                             |     |
                |     |                             |     |
                |     |     Protection Path #1      |     |
                |     |*****************************|     |
                |     |            ....             |     |
                |     |     Protection Path #m      |     |
                |     |*****************************|     |
                +-----+                             +-----+
                    |--------Protection Domain--------|
```

Figure 1: m:n protection domain

3. Use cases

3.1. m:1 (m > 1) protection

In the MPLS-TP linear protection such as 1+1/1:1 MPLS-TP protection, when a single failure is detected on the working path, the normal traffic can be restored to a protection path. The normal traffic however into a unprotected condition until the working path is completely repaired, that could put pressure on network operations.

The m:1 protection can increase service availability and reduce operator’s pressure, because it take multiple protection paths to ensuring high-priority services continue to operate on the 2nd, 3rd or N th alternate backup, at least one of m protection paths is an available.

The 2nd, 3rd or N th alternate backup paths may be provided in following cases.

3.1.1. pre-configuration

Before failure detection and/or notification, the protection relationship between the working and two or more protection paths SHOULD be configured and the protection path MUST be identified prior to use of the protection paths.

The unprotected extra traffic can be transported over the M protection path whenever the protection paths are not used to carry a normal traffic.

3.1.2. on-demand configuration

The protection relationship between a working path and a protection path are configured in the normal condition.

Other protection path such as 2nd, 3rd or N th alternate backup path is configured by either a control protocol or static configuration by the management system, only after failure detection and/or notification of either the working path or the protection path.

However, even when the configuration is performed by a control protocol, e.g. Generalized MPLS (GMPLS), the control protocol SHALL NOT be used as the primary mechanism for detecting or reporting network failures, or for initiating or coordinating protection switch-over. That is, it SHALL NOT be used as the primary resilience mechanism.

3.1.3. on-demand activation

Before failure detection and/or notification, two or more protection paths are instantiated between the same ingress-egress node pair as the working path, but note that the resources of m (m > 1) protection path may not be allocated.

The resource allocation on the m th protection path occurs only after failure detection and/or notification of either the working path or the protection path.

Therefore, this mechanism can against multiple failures but requires activation of the resource of m th protection path at ingress node and egress node after failure occurrence. After activated the m th protection path, the ingress node and egress node can carry the normal traffic.

3.2. m:n (m, n > 1) protection

In order to reduce backup costs, in the m:n architecture type, m dedicated protection transport paths are sharing backup resources for n working transport paths.

The bandwidth of each protection path should be allocated in such a way that it may be possible to protect any of the n working paths in case at least one of the m protection paths is available. When a working path is determined to be impaired, its normal user traffic signal first must be assigned to an available protection transport path followed by transition from the working to the assigned protection path at both the ingress node and egress node of the protected domain. It is noted that when more than m working paths are impaired, only m working paths can be protected

On the other hand, the normal traffic is either permanently connected to its working path and may be connected to one of the protection paths. It is noted that when at least one of the m protection paths is available, than the working path can be protected.

4. Requirements

Some recovery requirements are defined [RFC5654]. That however is limited to cover single failure and is not able to care that the multiple failures. This Section 4 extends the requirements to support the multiple failures scenarios.

MPLS-TP MUST support m:1 protection with the following requirements:

R1 The m:1 protection MUST protects against multiple failures that are detected on both of working path and protection path.

R2 The backup paths pre-configuration SHOULD be supported.

R3 On-demand backup paths configuration MAY be supported.

R4 On-demand backup resource activation MAY be supported.

R5 Some priority schemes MUST be provided, because a protection path has to choose between two or more backup resources.

MPLS-TP MUST support m:n protection with the following requirements:

R6 The m:n protection MUST protects against multiple failures that are simultaneously-detected on both of working path and protection path or more than one multiple working paths.

R7 Some priority schemes MUST be provided, because the backup resources are shared by multiple working paths dynamically.

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5654] Niven-Jenkins, B., Brungard, D., Betts, M., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, September 2009.

[RFC6378] Weingarten, Y., Bryant, S., Osborne, E., Sprecher, N., and A. Fulignoli, "MPLS Transport Profile (MPLS-TP) Linear Protection", RFC 6378, October 2011.

Authors’ Addresses

Zhenlong Cui
NEC

Email: c-sai@bx.jp.nec.com

Rolf Winter
NEC

Email: Rolf.Winter@neclab.eu

Himanshu Shah
Ciena

Email: hshah@ciena.com

Sam Aldrin
Huawei Technologies

Email: aldrin.ietf@gmail.com

Masahiro Daikoku
KDDI

Email: ms-daikoku@kddi.com

MPLS Working Group
INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: March 23, 2015

Santosh Esale
Raveendra Torvi
Chris Bowers
Juniper Networks

Luay Jalil
Verizon

U. Chunduri
Ericsson Inc.

Zhenbin Li
Huawei

September 19, 2014

Application-aware Targeted LDP
draft-esale-mpls-app-aware-tldp-01

Abstract

Recent targeted LDP applications such as remote loop-free alternates and BGP auto discovered pseudowire may automatically establish a tLDP session to any LSR in a network. The initiating LSR has information about the targeted applications to administratively control initiation of the session. However the responding LSR has no such information to control acceptance of this session. This document defines a mechanism to advertise and negotiate Targeted Applications Capability during LDP session initialization. As the responding LSR becomes aware of targeted applications, it may establish a limited number of tLDP sessions for certain applications. In addition, each targeted application is mapped to LDP Forwarding Equivalence Class (FEC) Elements to advertise only necessary LDP FEC-label bindings over the session.

Status of this Memo

Copyright and License Notice

Table of Contents

1 Introduction

LDP can use the extended discovery mechanism to establish a tLDP adjacency and subsequent session as described in [RFC5036]. An LSR initiates extended discovery by sending a tLDP Hello to a specific address. The remote LSR decides either to accept or ignore a tLDP Hello based on local configuration only. For an application such as FEC 128 pseudowire, the remote LSR is configured with the source LSR address, so the remote LSR can use that information to accept or ignore a given tLDP Hello.

Applications such as Remote LFA and BGP auto discovered pseudowire automatically initiate asymmetric extended discovery to any LSR in a network based on local state only. With these applications, the remote LSR is not explicitly configured with the source LSR address. so the remote LSR either responds to all LDP requests or ignores all LDP requests.

In addition, since the session is initiated and established after adjacency formation, the responding LSR has no targeted applications information to choose the targeted application it is configured to support. Also, the initiating LSR may employ a limit per application on locally initiated automatic tLDP sessions, however the responding LSR has no such information to employ a similar limit on the incoming tLDP sessions. Further, the responding LSR does not know whether the source LSR is establishing a tLDP session for a configured or an automatic application or both.

This document proposes and describes a solution to advertise Targeted Application Capability, consisting of a targeted application list, during initialization of a tLDP session. It also defines a mechanism to enable a new application and disable an old application after session establishment. This capability advertisement provides the responding LSR with the necessary information to control the acceptance of tLDP sessions per application. For instance, an LSR may accept all BGP auto discovered tLDP sessions but may only accept limited number of Remote LFA tLDP sessions.

Also, targeted LDP application is mapped to LDP FEC element type to advertise specific application FECs only, avoiding the advertisement of other unnecessary FECs over a tLDP session.

1.1 Terminology

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

2. Targeted Application Capability

An LSR MAY advertise that it is capable to negotiate a targeted LDP application list over a tLDP session by using the Capability Advertisement as defined in [RFC5561].

A new optional capability TLV is defined, ’Targeted Application Capability (TAC)’. Its encoding is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F| Targeted App. Cap.(IANA)|             Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|  Reserved   |                                               |
+-+-+-+-+-+-+-+-+                                               |
|                                                               |
~                 Targeted App. Cap. data                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
As described in [RFC5561]
U: set to 1. Ignore, if not known.
F: Set to 0. Do not forward.
S: MUST be set to 1 or 0 to advertise or withdraw the Targeted
   Application Capability TLV respectively.
```

Targeted Application Capability data:
A Targeted Applications Capability data consists of none, one or more 16 bit Targeted Application Elements. Its encoding is as follows:

Targeted Application Element(TAE)

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Targ. Appl. Id|E|  Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Targeted Application Identifier (TA-Id):

```
0x01: LDPv4 Tunneling
0x02: LDPv6 Tunneling
0x03: mLDP Tunneling
0x04: LDPv4 Remote LFA
0x05: LDPv6 Remote LFA
0x06: LDP FEC 128 Pseudowire
```

```
0x07: LDP FEC 129 Pseudowire
0x08: LDPv4 Session Protection
0x09: LDPv6 Session Protection
0x0A: LDP ICCP
0x0B: LDP P2MP PW
0x0C: mLDP node protection
```

E-bit: The enable bit indicates whether the sender is advertising or withdrawing the Targeted Application. The E-bit value is used as follows:

1 - The TAE is advertising the targeted application.
0 - The TAE is withdrawing the targeted application.

The length of TAC depends on the number of TAEs. For instance, if two TAEs are added, the length is set to 5. If both the peers advertise TAC, an LSR decides to establish or close a tLDP session based on the negotiated targeted application list.

For instance, suppose a initiating LSR advertises A, B and C as TA-Ids. Further, suppose the responding LSR advertises C, D and E as TA-Ids. Than the negotiated TA-Id, as per both the LSRs is C. In the second instance, suppose a initiating LSR advertises A, B and C as TA-Ids and the responding LSR, which acts as a passive LSR, advertises all the applications - A, B, C, D and E that it supports over this session. Than the negotiated targeted application as per both the LSRs are A, B and C. In the last instance, suppose the initiating LSR advertises A, B and C as a TA-Ids and the responding LSR advertises D and E as TA-Ids, than the negotiated targeted applciations as per both the LSRs is none. The Responding LSR sends 'Session Rejected/Targeted Application Capability Mis-Match' Notification message to the initiating LSR and may close the session.

3. Targeted Application Capability Procedures

At tLDP session establishment time, a LSR MAY include a new capability TLV, Targeted Application Capability (TAC) TLV, as an optional TLV in the LDP Initialization message. The TAC TLV's Capability data MUST consists of none, one or more Targeted Application Element(TAE) each pertaining to a unique Targeted Application Identifier(TA-Id) that a LSR supports over the session. If the receiver LSR receives the same TA-Id in more than one TAE, it MUST process the first element and ignore the duplicate elements. If the receiver LSR receives an unknown TA-Id in a TAE, it MUST silently ignore such a TAE and continue processing the rest of the TLV.

If the receiver LSR does not receive the TAC in the Initialization message or it does not understand the TAC TLV, the TAC negotiation MUST be considered unsuccessful and the session establishment MUST proceed as per [RFC5036]. On the receipt of a valid TAC TLV, an LSR MUST generate its own TAC TLV with TAEs consisting of unique TA-Ids that it supports over the tLDP session. If there is at least one TAE common between the TAC TLV it has received and its own, the session MUST proceed to establishment as per [RFC5036]. If not, A LSR MUST send a 'Session Rejected/Targeted Application Capability Mis-Match' Notification message to the peer and close the session. The initiating LSR playing the passive role in LDP session establishment MAY tear down the corresponding tLDP adjacency.

When the responding LSR playing the active role in LDP session establishment receives a 'Session Rejected/Targeted Application Capability Mis-Match' Notification message, it MUST set its session setup retry interval to a maximum value, as 0xffff. The session MAY stay in non-operational state. When it detects a change in the initiating LSR configuration or local LSR configuration pertaining to TAC TLV, it MUST clear the session setup back off delay associated with the session to re-attempt the session establishment. A LSR detects configuration change on the other LSR with the receipt of tLDP Hello message that has a higher configuration sequence number than the earlier tLDP Hello message.

When the initiating LSR playing the active role in LDP session establishment receives a 'Session Rejected/Targeted Application Capability Mis-Match' Notification message, either it MUST set its session setup retry interval to a maximum value, as 0xffff or it MUST teard down the corresponding tLDP adjacency with the session. This also leads to destruction of the session.

If it sets the session setup retry interval to maximum, the session MAY stay in a non-operational state. When this LSR detects a change in the responding LSR configuration or its own configuration pertaining to TAC TLV, it MUST clear the session setup back off delay associated with the session to re-attempt the session establishment.

If it decides to tear down the associated tLDP adjacency, the session is destroyed on the initiating as well as the responding LSR. The initiating LSR MAY take appropriate actions if it is unable to bring up the tLDP session. For instance, if an automatic session intended to support the Remote LFA application is rejected by the responding LSR, the initiating LSR may inform the IGP to calculate another PQ node [I-D.draft-ietf-rtgwg-remote-lfa] for the route or set of routes. More specific actions are a local matter and outside the scope of this document.

After a tLDP session has been established with TAC capability, the initiating and responding LSR MUST distribute FEC-label bindings for the negotiated applications only. For instance, if the tLDP session is established for BGP auto discovered pseudowire, only FEC 129 label bindings MUST be distributed over the session. Similarly, a LSR operating in downstream on demand mode MUST request FEC-label bindings for the negotiated applications only.

If the Targeted Application Capability and Dynamic Capability, as described in RFC 5561, are negotiated during session initialization, TAC MAY be re-negotiated after session establishment by sending an updated TAC TLV in LDP Capability message. The updated TLV MUST consist of one or more TAEs with E-bit set or E-bit off to advertise or withdraw the new and old application respectively. This may lead to advertisements or withdrawals of certain types of FEC-Label bindings over the session or tear down of the tLDP adjacency and subsequently the session.

The Targeted Application Capability is advertised on tLDP session only. If the tLDP session changes to link session, a LSR should withdraw it with S bit set to 0, which indicates wildcard withdral of all TAE elements. Similarly, if the link session changes to tLDP, a LSR should advertise it via the Capability message. If the capability negotiation fails, this may lead to destruction of the tLDP session.

Also, currently the remote LSR accepts asymmetric extended Hellos by default or by appropriate configuration. With this document, it should accept by default in order to then accept or reject the tLDP session based on the application information.

4. Interaction of Targeted Application Capabilities and State Advertisement Control Capabilities

As described in this document, the set of Targeted Application Elements negotiated between two LDP peers advertising TAC represents the willingness of both peers to advertise state information for a set of applications. The set of applications negotiated by the TAC mechanism is symmetric between the two LDP peers. In the absence of further mechanisms, two LDP peers will both advertise state information for the same set of applications.

As described in [I-D.draft-ietf-mpls-ldp-ip-pw-capability], State Advertisement Control(SAC) TLV can be used by an LDP speaker to communicate its interest or disinterest in receiving state information from a given peer for a particular application. Two LDP peers can use the SAC mechanism to create asymmetric advertisement of state information between the two peers for any particular application.

For a given tLDP session, the TAC mechanism can be used without the SAC mechanism, and the SAC mechanism can be used without the TAC mechanism. It is useful to discuss the behavior when TAC and SAC mechanisms are used on the same tLDP session. The TAC mechanism takes precedence over the SAC mechanism with respect to enabling applications for which state information will be advertised. For an tLDP session using the TAC mechanism, the LDP peers MUST NOT advertise state information for an application that has not been negotiated in the most recent Targeted Application Elements list (referred to as an un-negotiated application). This is true even if one of the peers announces its interest in receiving state information that corresponds to the un-negotiated application by sending a SAC TLV. In other words, when TAC is being used, SAC cannot enable state information advertisement for applications that have not been enabled by TAC.

On the other hand, the SAC mechanism takes precedence over the TAC mechanism with respect to disabling state information advertisements. If an LDP speaker has announced its disinterest in receiving state information for a given application to a given peer using the SAC mechanism, its peer MUST NOT send state information for that application, even if the two peers have negotiated that the corresponding application via the TAC mechanism.

For the purposes of determining the correspondence between targeted applications defined in this document and application state as defined in [I-D.draft-ietf-mpls-ldp-ip-pw-capability] an LSR MAY using the following mappings:

LDPv4 Tunneling - IPv4 Prefix-LSPs
LDPv6 Tunneling - IPv6 Prefix-LSPs
LDPv4 Remote LFA - IPv4 Prefix-LSPs
LDPv6 Remote LFA - IPv6 Prefix-LSPs
LDP FEC 128 Pseudowire - FEC128 P2P-PW
LDP FEC 129 Pseudowire - FEC129 P2P-PW
LDPv4 Session Protection - IPv4 Prefix-LSPs
LDPv6 Session Protection - IPv6 Prefix-LSPs

An LSR MAY map Targeted Application to LDP capability as follows:

mLDP Tunneling - P2MP Capability, MP2MP Capability

5. Targeted Application capability in LDP messages

5.1 TAC in LDP Initialization message

1. The S-bit of the Targeted Application Capability TLV MUST be set to 1 to advertise Targeted Application Capability and

SHOULD be ignored on the receipt.

2. The E-bit of the Targeted Application Element MUST be set to 1 to enable Targeted application.

3. An LSR MAY add State Control Capability by mapping Targeted Application Element to State Advertisement Control (SAC) Elements as defined in Section 4.

4. The LSR MAY add a different KeepAlive Time [RFC5036] value for an automatic tLDP session.

5.2 TAC in LDP Capability message

The initiating or responding LSR may re-negotiate the TAC after local configuration change with the Capability message.

1. The S-bit of Targeted Application Capability is set to 1 or 0 to advertise or withdraw it.

2. After configuration change, If there is no common TAE between its new TAE list and peers TAE list, the LSR MUST send a ’Session Rejected/Targeted Application Capability Mis-Match’ Notification message and close the session.

3. If there is a common TAE, a LSR MAY also update SAC Capability based on updated TAC as described in section 4 and sends the updated TAC and SAC capabilities in a Capability message to the peer.

4. A receiving LSR processes the Capability message with TAC TLV. If the S-bit is set to 0, the TAC is disbaled for the session. After that, the session may remain in established state or torn down based on [RFC5036] rules.

5. If the S-bit is set to 1, a LSR process a list of TAEs from TACs capability data with E-bit set to 1 or 0 to update the peers TAE. Also, it udpates the negotiated TAE list over the tLDP session.

6. Use cases

6.1 Remote LFA Automatic Targeted session

An LSR determines that it needs to form an automatic tLDP session to remote LSR based on IGP calculation as described in [I-D.draft-ietf-rtgwg-remote-lfa] or some other mechanism, which is outside the scope of this document. The LSR forms the tLDP adjacency and during session setup, constructs an Initialization message with Targeted Applications Capability (TAC) with Targeted Application Element (TAE) as Remote LFA. The receiver LSR processes the LDP Initialization message and verifies whether it is configured to accept a Remote LFA tLDP session. If it is, it may further verify that establishing such a session does not exceed the configured limit for Remote LFA sessions. If all these conditions are met, the receiver LSR may respond back with an Initialization message with TAC corresponding to Remote LFA, and subsequently the session may be established.

After the session has been established with TAC capability, the sender and receiver LSR distribute IPv4 or IPv6 FEC label bindings over the session. Further, the receiver LSR may determine that it does not need these FEC label bindings. So it may disable the receipt of these FEC label bindings by mapping targeted application element to state control capability as described in section 4.

6.2 FEC 129 Auto Discovery Targeted session

BGP auto discovery or other mechanisms outside the scope of this document MAY determine whether an LSR needs to initiate an auto-discovery tLDP session with a border LSR. Multiple LSRs MAY try to form an auto discovered tLDP session with a border LSR. So, a service provider may want to limit the number of auto discovered tLDP sessions a border LSR may accept. As described in Section 3, LDP may convey targeted applications with TAC TLV to border LSR. A border LSR may establish or reject the tLDP session based on local administrative policy. Also, as the receiver LSR becomes aware of targeted applications, it can also employ an administrative policy for security. For instance, it can employ a policy 'accept all auto-discovered session from source-list'.

Moreover, the sender and receiver LSR MUST exchange FEC 129 label bindings only over the tLDP session.

6.3 LDP over RSVP and Remote LFA targeted session

A LSR may want to establish a tLDP session to a remote LSR for LDP over RSVP tunneling and Remote LFA applications. The sender LSR may add both these applications as a unique Targeted Application Element in the Targeted Application Capability data of a TAC TLV. The receiver LSR MAY have reached a configured limit for accepting Remote LFA automatic tLDP sessions, but it may also be configured to accept LDP over RSVP tunneling. In such a case, the tLDP session is formed for both LDP over RSVP and Remote LFA applications as both needs same FECs - IPv4 and/or IPv6.

Also, the sender and the receiver LSR MUST distributes IPv4 and or IPv6 FEC label bindings only over the tLDP ession.

6.4 mLDP node protection targeted session

A merge point LSR may determines that it needs to form automatic tLDP session to the upstream point of local repair (PLR) LSR for MP2P and MP2MP LSP node protection as described in the [I-D.draft-ietf-mpls-mldp-node-protection] or other documents, which is outside the scope of this document. The MPT LSR may add a new targeted LDP application - mLDP node protection, as a unique TAE in the Targeted Application Capability Data of a TAC TLV and send it in the Initialization message to the PLR. If the PLR is configured for mLDP node protection and establishing this session does not exceed the limit of either mLDP node protection sessions or automatic tLDP sessions, the PLR may decide to accept this session. Further, the PLR responds back with the initialazation message with a TAC TLV that has one of the TAEs as - mLDP node protection and the session procceds to establishemt as per RFC 5036.

7 Security Considerations

The Capability procedure described in this document will apply and does not introduce any change to LDP Security Considerations section described in [RFC5036].

8 IANA Considerations

This document requires the assignment of a new code point for a Capability Parameter TLVs from the IANA managed LDP registry "TLV Type Name Space", corresponding to the advertisement of the Targeted Applications capability. IANA is requested to assign the lowest available value after 0x050B.

Value	Description	Reference
TBD1	Targeted Applications capability	[This draft]

This document requires the assignment of a new code point for a status code from the IANA managed registry "STATUS CODE NAME SPACE" on the Label Distribution Protocol (LDP) Parameters page, corresponding to the notification of session Rejected/Targeted Application Capability Mis-Match. IANA is requested to assign the lowest available value after 0x0000004B.

Value	Description	Reference

TBD2 Session Rejected/Targeted
Application Capability Mis-Match [This draft]

This document also creates a new name space 'the LDP Targeted Application Element type' on the Label Distribution Protocol (LDP) Parameters page, that is to be managed by IANA. The range is 0-255, with the following values requested in this document.

0x00: Reserved
0x01: LDPv4 Tunneling
0x02: LDPv6 Tunneling
0x03: mLDP Tunneling
0x04: LDPv4 Remote LFA
0x05: LDPv6 Remote LFA
0x06: LDP FEC 128 Pseudowire
0x07: LDP FEC 129 Pseudowire
0x08: LDPv4 Session Protection
0x09: LDPv6 Session Protection
0x0A: LDP ICCP
0x0B: LDP P2MP PW
0x0C: mLDP node protection
0xFF: Reserved

The allocation policy for this space is 'Standards Action'.

9. Acknowledgments

The authors wish to thank Nischal Sheth, Hassan Hosseini, Kishore Tiruveedhula, Kamran Raza and Loa Andersson for doing the detailed review. Thanks to Manish Gupta and Martin Ehlers for their input to this work and for many helpful suggestions.

10 References

10.1 Normative References

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, October 2007.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, July 2009.

[I-D.draft-ietf-mpls-ldp-ip-pw-capability] Kamran Raza, Sami Boutros, "Disabling IPoMPLS and P2P PW LDP Application's State Advertisement", draft-ietf-mpls-ldp-ip-pw-capability-07 (work in progress), April 27, 2014.

[I-D.draft-ietf-mpls-mldp-node-protection] IJ. Wijnands, E. Rosen, K. Raza, J. Tantsura, A. Atlas, Q. Zhao, "mLDP Node Protection", draft-ietf-mpls-mldp-node-protection-01 (work in progress), February 13, 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2 Informative References

[I-D.draft-ietf-rtgwg-remote-lfa] S. Bryant, C. Filsfils, S. Previdi,M. Shand, "Remote LFA FRR", draft-ietf-rtgwg-remote-lfa-06 (work in progress), May 23, 2014.

[RFC6074] E. Rosen, B. Davie, V. Radoaca, and W. Luo, "Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)"

[RFC4762] M. Lasserre, and V. Kompella, "Virtual Private LAN Service VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, January 2007.

[RFC4447] L. Martini, E. Rosen, El-Aawar, T. Smith, and G. Heron, "Pseudowire Setup and Maintenance using the Label Distribution Protocol", RFC 4447, April 2006.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, August 2008.

Authors’ Addresses

Santosh Esale
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US
EMail: sesale@juniper.net

Raveendra Torvi
Juniper Networks
10 Technology Park Drive.
Westford, MA 01886
US

EMail: rtorvi@juniper.net

Chris Bowers
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US
EMail: cbowers@juniper.net

Luay Jalil
Verizon
1201 E Arapaho Rd.
Richardson, TX 75081
US
Email: luay.jalil@verizon.com

Uma Chunduri
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA
Email: uma.chunduri@ericsson.com

Zhenbin Li
Huawei Bld No.156 Beiqing Rd.
Beijing 100095
China
Email: lizhenbin@huawei.com

INTERNET-DRAFT
Intended Status: Standards Track
Expires: April 30, 2015

Luyuan Fang
Vijay Gill
Microsoft
Fabio Chiussi

October 27, 2014

MPLS-Based Hierarchical SDN for Hyper-Scale DC/Cloud
draft-fang-mpls-hsdn-for-hsdc-00

Abstract

This document describes Hierarchical SDN (HSDN), an architectural solution to scale the Data Center (DC) and Data Center Interconnect (DCI) networks to support tens of millions of physical underlay endpoints, while efficiently handling both ECMP and any-to-any end-to-end Traffic Engineered (TE) traffic. HSDN achieves massive scale using surprisingly small forwarding tables in the network nodes and brings key simplifications in the control plane as well. The HSDN forwarding architecture is based on four main concepts: 1. Dividing the DC and DCI in a hierarchically-partitioned structure; 2. Assigning groups of Underlay Border Nodes in charge of forwarding within each partition; 3. Constructing HSDN MPLS label stacks to identify the endpoints according to the HSDN structure; and 4. Forwarding using the HSDN MPLS labels.

Status of this Memo

http://www.ietf.org/shadow.html

Copyright and License Notice

Table of Contents

1. Introduction

With the growth in the demand for cloud services, the end-to-end cloud network, which includes Data Center (DC) and Data Center Interconnect (DCI) networks, has to scale to support millions to tens of millions of underlay network endpoints. These endpoints can be bare metal servers, virtualized servers, or physical and virtualized network functions and appliances.

The scalability challenge is twofold: 1. Being able to scale using low-cost network nodes while achieving high resource utilization in the network; and 2. Being able to scale at low operational and computational complexity while supporting Equal-Cost Multi-Path (ECMP) and any-to-any Traffic Engineering (TE).

An important set of scalability issues to resolve comes from the potential explosion of the routing tables in the network nodes as the number of underlay network endpoints increases. Current commodity switches have relatively small routing and forwarding tables. For example, the typical Forwarding Information Base (FIBs) and Label Forwarding Information Base (LFIBs) tables in current low-cost network nodes contain 16K or 32K entries. These small sizes are clearly insufficient to support entries for all the endpoints in the hyper-scale cloud. Address aggregation is used to ameliorate the problem, but the scalability challenges remain, since the dynamic and elastic environment in the DC/cloud often brings the need to handle finely granular prefixes in the network.

Other factors contribute to the FIB/LFIB explosion. For example, in a typical DC using a fat Clos topology, even the support of ECMP load balancing may become an issue if the individual outgoing paths belonging to an ECMP group carry different outgoing labels.

Another key scalability issue to resolve is the complexity of certain desired functions that should be supported in the network, the most prominent one being TE. Currently, any-to-any server-to-server TE in the DC/DCI is simply unfeasible, as path computation and bandwidth allocation at scale, an NP-complete problem, becomes rapidly unmanageable. Furthermore, the forwarding state needed in the network nodes for TE tunnels contributes in a major way to the explosion of the LFIBs.

Other major scalability issues are related to the efficient creation, management, and use of tunnels, for example the configuration of protection paths for fast restoration.

Many additional scalability issues in terms of operational and computational complexity need to be resolved in order to scale the control plane and the network state. In particular, the controller-centric approach of Software Defined Networks (SDNs), which is increasingly being accepted as "the way to build the next generation clouds," in order to be scalable, requires appropriate scalability solutions in order to take full advantage of the potential benefits of SDN.

Finally, the underlay network architecture should offer certain capabilities to facilitate the support of the demand of the overlay network.

In this document, we present Hierarchical SDN (HSDN), a set of solutions for all these scalability challenges in the underlay network, both in the forwarding and in the control plane. Although HSDN can be used in principle with any forwarding technology, it has been designed to leverage Multi Protocol Label Switching (MPLS)-based forwarding [RFC3031], using label stacks [RFC3032] constructed according to the HSDN structure.

HSDN achieves massive scale using surprisingly small LFIBs in the network nodes, while supporting both ECMP and any-to-any end-to-end TE traffic. HSDN also brings important simplifications in the control plane and in the architecture of the SDN controller.

The HSDN forwarding architecture is based on four main concepts: 1. Dividing the DC and DCI in a hierarchically-partitioned structure; 2. Assigning groups of Underlay Border Nodes in charge of forwarding within each partition; 3. Constructing HSDN MPLS label stacks to identify the end points according to the HSDN structure; and 4. Forwarding using the HSDN MPLS labels.

HSDN is designed to allow the physical decoupling of the control and forwarding, and have the LFIBs configured by a controller according to the SDN approach. However, it is also meant to support the traditional distributed routing and label distribution protocol approach, which may be particularly useful during technology migration.

1.1. Terminology

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

Term	Definition
BGP	Border Gateway Protocol
DC	Data Center
DCGW	DC Gateway (Border Leaf)
DCI	Data Center Interconnect
DID	Destination Identifier
ECMP	Equal Cost MultiPathing
FIB	Forwarding Information Base
HSDN	Hierarchical SDN
LDP	Label Distribution Protocol
LFIB	Label Forwarding Information Base
LN	Leaf Node
MPLS	Multi-Protocol Label Switching
PID	Path Identifier
SDN	Software Defined Network
SN	Spine Node
SVR	Server
UP	Underlay Partition
UPBG	Underlay Partition Border Group
UPBG	Underlay Partition Border Node
TE	Traffic Engineering
ToR	Top-of-Rack switch
TR	Top-of-Rack switch
VN	Virtual Network
VM	Virtual Machine
WAN	Wide Area Network

In this document, we also use the following terms.

- End device: A physical device attached to the DC/DCI network. Examples of end devices include bare metal servers, virtualized servers, network appliances, etc.

- Level: A layer in the hierarchy of underlay partitions in the HSDB architecture.

- Overlay Network (ON): A virtualized network that provides Layer 2 or Layer 3 virtual network services to multiple tenants. It is implemented over the underlay network.

- Path Label (PL): A label used for MPLS-based HSDN forwarding in the underlay network.

- Row: A row of racks where end devices reside in a DC.

- Tier: One of the layers of network nodes in a multi-layer Clos-based topology.

- o Underlay Network (UN): The physical network that provides the connectivity among physical end devices. It provides transport for the overlay network traffic.

- o Underlay Partition (UP): A logical portion of the underlay network designed according to the HSDN architecture. Underlay partitions are arranged in a hierarchy consisting of multiple levels.

- o VN Label (VL): A label carrying overlay network traffic. It is encapsulated in the underlay network in a stack of path labels constructed according to the HSDN forwarding scheme.

1.2. DC and DCI Reference Model

Here we show the typical structure of the DC and DCI, which we use in the rest of this document to describe the HSDN architecture. We also introduce a few commonly used terms to assist in the explanation.

Figure 1 illustrates multiple DCs interconnected by the DCI/WAN.

```
                     +-------------+
                     |             |
                     |     DC      |
                     |             |
                     +-------------+
                               \-----.
                               (       ')
                           .--(.       '.---.
                          (     ,      ,     )
                         (       DCI/WAN      )
                          (.                .)
                           (     (        .)  \
                            /'--' '-''---'  +-------------+
                 +-------------+            |             |
                 |             |            |     DC      |
                 |     DC      |            |             |
                 |             |            +-------------+
                 +-------------+
```

Figure 1. DCIWAN interconnecting multiple DCs.

Figure 2 below illustrates the typical structure of a Clos-based DC fabric.

```
                                 +----+   +----+
  DCGW                 +---------+ GW +   + GW +---------+
                       |         ++--++\ /++--++         |
                       | +--------|--|--\--\  |          |
                       | |        |  \-/-\-|--|--------+ |
                     +-+-++     +-+--+/   \+--+-+     ++-+-+
  Spine    +---------+ SN +     | SN |     | SN +     + SN +---------+
           |         +----+\    ++--++     ++--++\   /+----+         |
           | +---------------\---+  |       |  +---/---------------+ |
           | |        ...      \    |       |    /      ...        | |
          ++-+-+ +----+  +----+ ++--++     ++--++ +----+  +----+ +-+-++
  Leaf    | LN | | LN |  | LN | | LN |     ||LN | | LN |  | LN | | LN |
          ++--++ ++--++  +---++ ++---+     +---++ ++--++  +---++ ++--++
           |   \ /   |    |   \ /   |       |   \ /   |    |   \ /   |
           |   / \   |    |   / \   |       |   / \   |    |   / \   |
           | / ... \ |    | / ... \ |       | / ... \ |    | / ... \ |
         ++++ +--+ ++++ ++++ +--+ ++++    ++++ +--+ ++++ ++++ +--+ ++++
  ToR    |TR| |TR| |TR| |TR| |TR| |TR|    |TR| |TR| |TR| |TR| |TR| |TR|
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
          |    |    |    |    |    |       |    |    |    |    |    |
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
  Server +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
  rack   +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
```

Figure 2. Typical Clos-based DC fabric topology.

Note: Not all links are shown in Figure 2.

The DC fabric shown in Figure 2 uses what is known as a spine and leaf architecture with a multi-stage Clos-based topology interconnecting multiple tiers of network nodes. The DC Gateways (DCGWs) connect the DC to the DCI/WAN. The DCGW connect to the Spine Nodes (SNs), which in turn connect to the Leaf Nodes (LFs). The Leaf Nodes connect to the Top-of-Rack switches (ToRs). Each ToR typically resides in a rack (hence the name) accommodating a number of servers connected to their respective ToR. The servers may be bare metal or virtualized.

Each tier of switches and the connectivity between switches is designed to offer a desired capacity and provide sufficient bandwidth to the servers and end devices.

The precise topology and connectivity between the tiers of switches depends on the specific design of the DC. More or less tiers of switches (spines or leaves) or asymmetric topologies, not shown in the figure, may be used. A precise description of the topology and its design criteria is out of the scope of this document.

What’s relevant for this document is the fact that a typical large-scale DC topology does not have all the tiers fully connected to the adjacent tiers. In other words, not all network nodes in a tier are connected to all the network nodes in the adjacent tiers. This is especially true for the tiers closer to the endpoints, and is due to the sheer number of connections and devices and the physical constraints of the DC, which makes it impractical, uneconomical, and ultimately unnecessary to use a fully connected Clos-based topology.

The connectivity is typically organized following an aggregation/multiplexing connectivity architecture that consolidates traffic from the edges into the leafs and spines.

2. Requirements

2.1. MPLS-Based HSDN Design Requirements

The following are the key design requirements for HSDN solutions.

1) MUST support millions to tens of millions of underlay network endpoints in the DC/DCI.

2) MUST use very small LFIB sizes (e.g., 16K or 32K LFIB entries) in all network nodes.

3) MUST support both ECMP and any-to-any, end-to-end, server-to-server TE traffic.

4) MUST support ECMP traffic load balancing using a single forwarding entry in the LFIBs per ECMP group.

5) MUST require IP lookup only at the network edges.

6) MUST support encapsulation of overlay network traffic, and support any network virtualization overlay technology.

7) MUST support control plane using both SDN controller approach, and the traditional distributed control plane approach using any label distribution protocols.

2.2. Hardware Requirements

The following are the hardware requirements to support HSDN.

1) The server NICs MUST be able to push a HSDN label stack consisting of as many path labels as levels in the HSDN hierarchical partition (e.g., 3 path labels).

2) The network nodes MUST support MPLS forwarding.

3) The network nodes MUST be able perform ECMP on packets carrying a label stack consisting of as many path labels as levels in the HSDN hierarchical partition, plus one or more VN label/header for the overlay network (e.g., 3 path labels + 1 VN label/header).

3. HSDN Architecture - Forwarding Plane

As mentioned above, a primary design requirement for HSDN is to enable scalability of the forwarding plane to tens of millions of network endpoints using very small LFIB sizes in all network nodes in the DC/DCI, while supporting both ECMP and any-to-any server-to-server TE traffic.

The driving principle of the HSDN forwarding plane is "divide and conquer" by partitioning the forwarding task into local and independent forwarding. When designed properly, such an approach enables extreme horizontal scaling of the DC/DCI.

HSDN is based on four concepts:

1) Dividing the underlay network in a hierarchy of partitions;
2) Assigning groups of Underlay Partition Border Nodes (UPBN) to each partition, in charge of forwarding within the corresponding partition;
3) Constructing HSDN label stacks for the endpoint Forward Equivalency Classes (FECs) in accordance with the underlay network partition hierarchy;
4) Configuring the LFIBs in all network nodes and forwarding using the label stacks.

In this section, we explain in detail each of these concepts. Scalability analysis for both ECMP and TE is presented in Section 4. In Section 5, we describe a possible label stack assignment scheme for HSDN.

3.1. Hierarchical Underlay Partitioning

HSDN is based on dividing the DC/DCI underlay network into logical partitions arranged in a multi-level hierarchy.

The HSDN hierarchical partitioning is illustrated in Figure 3.

```
           +----------------------------------------------------------+
  UP       |                          UP0                               |
  Level 0  |                                                          |
           |+---------------------------------+   +------------------+|
           ||                                 |...|                  ||
           +|---------------------------------|---|------------------|+
  UP        |             UP1-0               |   |       UP1-N      |
  Level 1   |                                 |   |                  |
            |+-------------+   +-------------+|   |   +-------------+|
            ||             |...|             ||   |...|             ||
            +|-------------|---|-------------|+   +---|-------------|+
  UP         |   UP2-0-0   |   |   UP2-0-N   |        |  UP2-N-N    |
  Level 2    |             |   |             |        |             |
             +-----+ +-----+   +-----+ +-----+        +-----+ +-----+
             | SVR | | SVR |   | SVR | | SVR |        | SVR | | SVR |
             |-----|-|-----|   |-----|-|-----|        |-----|-|-----|
  Overlay    |VM|VM| |VM|VM|   |VM|VM| |     |        |VM|VM| |     |
  Level      |-----| |-----|   |-----| |     |        |-----| |     |
             |VM|VM| |VM|VM|   |VM|VM| |     |        |VM|VM| |     |
             +-----+ +-----+   +-----+ +-----+        +-----+ +-----+
```

Figure 3. HSDN underlay network hierarchical partitioning of DC/DCI.

The hierarchy consists of multiple levels of Underlay Partitions (UPs). For simplicity, we describe HSDN using three levels of partitioning, but more or less levels can be used, depending on the size and architecture of the overall network, using similar design principles (as shown below, three levels of partitions are sufficient to achieve scalability to tens of millions servers using very small LFIBs).

The levels of partitions are nested into a hierarchical structure. At each level, the combination of all partitions covers the entire DC/DCI topology. In general, within each level, the UPs do not overlap, although there may be design scenarios in which overlapping UPs within a level may be used. The top level (Level 0) consists of a single underlay partition UP0 (the HSDN concept can be extended to multi-partitioned Level 0).

We use the following naming convention for the UPs:

- Partitions at Level i are referred to as UPi (e.g., UP0 for Level 0, UP1 for Level 1, UP2 for Level2, and so on).

- Within each level, partitions are identified by a rightmost sequential number (starting from 1) referring to the corresponding level and a set of sequential number(s) for each partition in a higher level that the specific partition is nested into.

For example, at Level 1, there are N partitions, referred to as UP1-1 to UP1-N.

Similarly, at Level 2, there are M partitions for each Level 1 partitions, for a total of NxM partitions. For example, the Level 2 partitions nested into Level 1 partition UP1-1 are UP2-1-1 to UP2-1-M, while the ones nested into UP1-N are UP2-N-1 to UP2-N-M.

- Note that for simplicity in illustrating the partitioning, we assume a symmetrical arrangement of the partitions, where the number of partitions nested into each partition at a higher level is the same (e.g., all UP1 partitions have M UP2 partitions). In practice, this is rarely the case, and the naming convention can be adapted accordingly for different numbers of partitions.

The following considerations complete the description of Figure 3.

o The servers (bare metal or virtualized) are attached to the bottom UP level (in our case, Level 2). A similar naming convention as the one used for the partitions may be used.

o In Figure 3, we also show an additional Overlay Level. This corresponds to the virtualized overlay network (if any) providing Virtual Networks (VN) connecting Virtual Machines (VMs) and other overlay network endpoints. Overlay network traffic is encapsulated by the HSDN underlay network. The operation of the Overlay Level is out of scope of this document.

The UPs are designed to contain one or more tiers of switches in the DC topology or nodes in the DCI. The key design criteria in defining the partitions at each layer is that they need to follow the "natural" connectivity implemented in the DC/DCI topology. An example is given below to further clarify how the partitions are designed.

3.2. Underlay Partition Border Nodes

Once the HSDN hierarchical partitioning is defined, Underlay Partition Border Nodes (UPBNs) are assigned to each UP. This is illustrated in Figure 4.

```
  +++++++++++++++++++++++++++++++++++++++++++++++++
  |                     UP0              <--------|-----+
  |                                               |     |
  | +++++++++++++++++++++++++++++++++++++++++++++ |     |
  | |   +-----------------------------------+   | |     |
  | |   | +---------+  UPBG1-i  +---------+ |   | |     |
  | |   | | UPBN1-i |    ...    | UPBN1-i | |   | |     |
  | |   | +---------+           +---------+ |   | |     |
  | |   +-----------------------------------+   | |     |
  ++|+++++++++++++++++++++++++++++++++++++++++++|++     |
    |                   UP1                <------|----+  |   +---------+
    | +++++++++++++++++++++++++++++++++++++++++ |    |  +---|   LP0   |
    | | +-----------------------------------+ | |    |      +---------+
    | | | +---------+ UPBG2-i-j +---------+ | | |    +------|   LP1   |
    | | | |UPBN2-i-j|    ...    |UPBN2-i-j| | | |           +---------+
    | | | +---------+           +---------+ | | |    +------|   LP2   |
    | | +-----------------------------------+ | |    |      +---------+
    ++|+++++++++++++++++++++++++++++++++++++++|++    |  +---|   VL    |
      |                 UP2              <----|------+  |   +---------+
      |          +---------------+            |         |
      |          |   SVR-i-j-k   |            |         |
      |          | +-----------+ |            |         |
      |          | |    NVE    |<-------------|---------+
      |          | +-----+-----+ |            |
      +++++++++++| | VM  | VM  | |+++++++++++++
                 | +-----+-----+ |
                 | | VM  | VM  | |
                 +-+-----+-----+-+
```

Figure 4. UBPNs, UBPGs, and label stack assignment.

The UPBNs belong to two partitions in adjacent levels in the hierarchy and they constitute the entry points for traffic from the higher level partition destined to the corresponding lower level partition. As such, they constitute the forwarding end destinations within each partition.

In order to provide sufficient capacity and support traffic load balancing between the levels in the hierarchy, multiple UPBNs are assigned to each partition. The UPBNs for each partition are grouped into an Underlay Partition Border Group (UPBG). As shown below, using an appropriate Label Stack Assignment scheme all UPBNs in a UPBG can be made identical for ECMP traffic forwarding (i.e., the ECMP entries in the LFIBs in all UPBNs in a UPBG are identical). Thus, for ECMP traffic load balancing, all UPBNs belong to the same FEC as far as the higher level partition is concerned. For TE traffic, a desired UPBN within a UPBG group may need to be specified, and thus the UPBNs in a UPBG are not forwarding-wise equivalent.

In practice, the UPs are designed by finding the most advantageous way to partition the DC Clos-based topology and the DCI topology. Within the DC, the UPBNs in each level are subsets of the network nodes in one of the tiers that form the multi-stage Clos architecture. In general, the UPs may internally contain tiers of network nodes that are not UPBNs. A specific design example to further illustrate the HSDN partitioning is provided below.

As explained in more detail below, for forwarding purposes, by partitioning the DC/DCI in this manner and using HSDN forwarding, the UPBNs need to have entries in their LFIBs only to reach destinations in the two partitions to which they belong to (their own corresponding partition and the higher-layer partition to which they nested to). The network nodes inside the UPs only need to have entries in their LFIB to reach the destinations in their partition.

From these considerations, a first design heuristic for choosing the partitioning structure is to keep the number of partitions nested at each level into the higher level relatively small for all levels. For the lowest level, the number of endpoints (servers) in each partition should also be kept to manageable levels. Clearly, the design tradeoff is between the size and the number of partitions at each level. Fortunately, for most practical deployments, it is relatively simple to find a good tradeoff that achieves the desired scalability.

3.2.1. UPBN and UPBG Naming Convention

We use a similar naming convention for the UPBNs and UPBGs as the one used for the UPs:

- UPBNi is a UPBN between partitions at Level(i) and Level(i-1). Similarly for UPBG.

- Within each level, the UPBNs are identified by a set of sequential number(s) equal to the corresponding sequential number(s) of the corresponding partition within that level.

  For example, at Level 1, UPBN1-1 corresponds to partition UP1-1, and connects UP0 with UP1-1. UPBN1-N corresponds to partition UP1-N and connects UP0 with UP1-N, and so on. Similarly for UPBG.

  At Level 2, UPBN2-1-1 corresponds to partition UP2-1-1 and connects UP1-1 with UP2-1-1, and so on. Similarly for UPBG.

  Note that the UPBNs within an UPBGs can be further distinguished using an appropriate naming convention, which is not shown here.

3.2.2. HSDN Label Stack

In MPLS-Based HSDN, an MPLS label stack is defined and used for forwarding. The key notion in HSDN is that the label stack is defined and the labels are assigned in accordance with the hierarchical partitioned structure defined above.

The label stack, shown in Figure 4 above, is constructed as follows.

- The label stack contains as many Path Labels (PLs) as levels in the partition hierarchy.

- Each PL in the label stack is associated to a corresponding level in the partition hierarchy and is used for forwarding at that level.

  In the scenario of Figure 4, PL0 is associated to Level 0 and is used to forward to destination in UP0, PL1 is associated to Level 1 and is used to forward to destinations in any UP1 partitions, and PL2 is associated to Level 2 and is used to forward to destinations in any UP2 partitions.

- A VN Label (VL) is also shown in the label stack in Figure 4. This label is associated to the Overlay Level and is used to forward in the overlay network. The VL is simply encapsulated in the label stack and transported in the HSDN underlay network. The details of the VL processing within the overlay network are out of scope of this document.

Each endpoint in the DC/DCI is identified by a corresponding label stack. For a given endpoint, the label stack is constructed in such a way that the PLO specifies the UP1 to which the endpoint is attached to, the PL1 specifies the UP1 to which the endpoint is attached to, and the PL2 specifies the FEC in the UP2 corresponding to the endpoint.

A scheme to assign the PL labels in the HSDN label stack is described below.

3.2.3. HSDN Design Example

We use an example to further explain the HSDN design criteria to define the hierarchically-partitioned structure of the DC/DCI. We use the same design example in the Scalability Analysis section below to show the LFIB sizing with ECMP and TE traffic.

To summarize some of the design heuristics for the HSDN underlay partitions:

- The UPs should be designed to follow the "natural" connectivity topology in the DC/DCI.

- The number of partitions at each level nested into the higher level should be relatively small (since they are FEC entries in the LFIBs in the network nodes in the corresponding levels).

- The number of endpoints (servers) in each partition in the lowest level should be relatively small (since they are FEC entries in the LFIBs in the network nodes in the lowest level).

- The number of levels should be kept small (since it corresponds to the number of path labels in the stack).

- The number of tiers in each partition in each level should be kept small. This is due to the multiplicative fanout effect for TE traffic (explained below), which has a major impact on the LFIB size needed to support any-to-any server-to-server TE.

The HSDN forwarding plane design consists in finding the best tradeoff among these contrasting objectives. Although the optimal design choices ultimately depend on the specific deployment, here we describe an illustrative example.

As shown below, a three-level HSDN hierarchy is sufficient to scale the DC/DCI to tens of millions of servers.

With three levels, a possible design choice for the UP1s is to have each UP1 correspond to a DC. With this choice, the UP0 corresponds to the DCI and the UPBN1s are the DCGWs in each DC (the UPBG1s group the DCGWs in each DC).

Once the UP1s are chosen this way, a possible design choice for the UP2s is to have each UP2 correspond to a group of racks, where each group of racks may correspond to a portion of a row of racks, an entire row of racks, or multiple rows of racks. The specific best choice of how many racks should be in a group of racks corresponding to each UP2 ultimately depends on the specific connectivity, the number of servers per racks.

While precise numbers depend on the specific technologies used in each deployment, here and in the Scalability Analysis section below we want to give some ideas of the scaling capabilities of HSDN. For this purpose, we use some hypothetical yet reasonable numbers to characterize the partitioning design example.

Assume the following: a) 20 DCs connected via the DCI/WAN; b) 50 servers per rack; c) 20 racks per group of racks; d) 50 groups of racks per DC.

With these numbers, there are 500K servers per DC, for a total of 10M underlay network endpoints in the DC/DCI.

In the HSDN structure in this example, there are 20 UP1s, 500 UP2s per UP1, and 1000 servers per UP2.

3.3. MPLS-Based HSDN Forwarding

The hierarchically partitioned structure and the corresponding label stack are used in HSDN to scale the forwarding plane horizontally while using LFIBs of surprising small sizes in the network nodes.

As explained above, each label in the HSDN label stack is associated with one of the levels in the hierarchy and is used to forward to destinations in the underlay partitions at that level.

We describe the life of a packet in the HSDN DC/DCI. We use the specific design example described in Section 3.2.3 above to help in the explanation, but of course the forwarding would be similar for other design choices.

We first describe the behavior for non-TE traffic. In the HSDN DC/DCI, for a packet that needs to be forwarded to a specific endpoint in the underlay network, the outer label PL0 specifies which UP1 contains the endpoint. Let’s refer to this UP1 as UP1-a. For ECMP traffic, the PL0 binding is with a FEC corresponding to the UPBG1-a associated with UP1-a. Note that all the endpoints reachable via UP1-a are forwarded using the same FEC entry for Level 0 in the hierarchical partitioning.

Once the packet reaches one of the network nodes UPBN1-a in the UPBG1-a group (the upstream network nodes perform ECMP load balancing, thus the packet may enter UP1-a via any of the UPBN1-a nodes), the PL0 is popped and the PL1 is used for forwarding in the UP1-a. To be precise, because of penultimate hop popping, it is the network node immediately upstream of the chosen UPBN1-a that pops the label P0).

The PL1 is used within UP1-a to reach the UP2 which contains the endpoint. Let’s refer to this UP2 as UP2-a. In the UP2 network nodes the PL1 binding is with a FEC corresponding to the UPBG2-a associated with UP2-a. Similarly as above, note that all the endpoints reachable via UP2-a are forwarded using the same FEC entry for Level 1 in the hierarchical partitioning.

Once the packet reaches one of the network nodes UPBN2-a in the UPBG2-a group (once again, the upstream network nodes perform ECMP load balancing, so the packet may transit to any of the UPBN2-a nodes), the PL1 is popped and the PL2 is used for the rest of the forwarding (again, to be precise, the penultimate network node upstream of UPBN2-a is the one popping the PL1 label).

The PL2 is used within UP2-a to reach the desired endpoint. Note that the UPBN2 nodes and the network nodes in the UP2s have entries in their LFIBs only to reach endpoints within their UP2. They can reach endpoints in other UP2s by using a FEC entry corresponding to the UP2 containing the destination endpoint, identified by PL1.

The following two observations help in further clarifying the forwarding operation above.

- The PL0 is used for forwarding from the source to the UPBN1-a. For a packet originating from an endpoint attached to a certain UP2, say UP2-b, nested to a different UP1, say UP1-b, PL0 is used for forwarding in all network nodes that the packet transits until it reaches the UPBN1-a. This includes network nodes in UP2-b and UP1-b (i.e., "on the way out" from UP2). It also includes one of the UPBN1-b nodes. Note, however, that the PL0 is not popped at the UPBN1-b, since it is used for forwarding to the destination UPBN1-a.

- Not all packets carry a three-label MPLS stack. For example, a packet originating from the endpoint in UP2-b and destined to an endpoint in the same UP2-b only carries PL2. Similarly a packet originating from the endpoint in UP2-b and destined to an endpoint in a different UP2 nested in the same UP1-b only carries PL1 and PL2.

In the case of TE traffic, the use of the different labels in the label stack is similar as what described above for ECMP traffic. However, the labels are bound to FECs identifying a specific path within each UPs that is traversed. Therefore, TE traffic contributes for additional entries in the LFIBs in the network nodes. By properly designing the UPs, the number of LFIB entries can be kept relatively small.

HSDN, by superimposing a hierarchically-partitioned structure and using a label stack constructed according to such a structure, is able to impose a forwarding scheme that is aggregated by construction. This translates in dramatic reductions in the size of the LFIBs in the network nodes, since each node only needs to know a limited portion of the forwarding space.

HSDN supports any label assignment scheme to generate the labels in the label stack. However, if a label assignment scheme that is consistent with the HSDN structure is used, additional simplifications of the LFIBs and the control plane can be achieved.

In Section 5 below, we present one example of such a scheme, where the labels in the label stack represent the "physical" location of the endpoint, expressed according to the HSDN structure. For TE traffic, the labels represent a specific path towards the desired destination through the HSDN structure.

In the Scalability Analysis section and in the Control Plane section below we assume that such a Label Assignment scheme is used.

In HSDN, the LFIBs in the network nodes can be configured in such a way that all the paths in the DC/DCI are pre-established. This can be achieved using surprisingly small LFIB sizes.

4. Scalability Analysis

In this section, we compute the maximum size of the LFIBs for non-TE/ECMP traffic and any-to-any server-to-server TE traffic.

4.1. LFIB Sizing - ECMP

For ECMP traffic, at each level, all destinations belonging to the same partition at a lower level are forwarded using the same FEC entry in the LFIB, which identifies the destination UPBG for that level, or the destination endpoint at the lower level. Since the UPs are designed in such a way to keep the number of destinations small in all UPs, and the network nodes only need to know how to reach destinations in their own UP and in the adjacent UP at the higher level in the hierarchy, this translate to the fact that hyper scale of the DC/DCI can be achieved with very small LFIB sizes in the all individual network nodes.

The worst case for the LFIB size occurs at one of the network nodes that serve as UPBNs for one of the levels of UPs in the hierarchy. The level where the LFIB size occurs depend on the specific choice of the partitioning design.

To be completed.

4.2. LFIB Sizing - TE

As noted above, TE traffic may add a considerable number of entries to LFIB, since it creates one new FEC per TE tunnel to each destination.

HSDN provides a solution to this problem. In fact, HSDN can support any-to-any server-to-server "TE Max Case" with small LFIB sizes. In TE Max Cases, all sources are connected to all destinations (e.g., server to server) with TE tunnels, the tunnels using all possible distinct paths in the network. TE Max Case gives therefore an upper bound to the number of TE tunnels (and consequently, LFIB entries) in the network.

Again, in HSDN, since the UPs are designed in such a way to be relatively small, the number of paths in each partition can be kept to a manageable number.

In a Clos Topology (the analysis can be extended to generic topologies), the number of paths in a UP with N destination can be easily computed. The number of paths (and the maximum number of LFIB entries is equal to the products of the switch fanout in each tier traversed from the source to the destination in that UP. We refer to this as the TE Fanout Multiplicative Effect, which is illustrated in Figure 5.

Total # LFIB Entries for TE Max Case = N * F1 * F2 * ... * F(M-1)

Where Fi is the fanout of a switch in each tier traversed to the destination, M is the number of tiers in the UP, and N is the number of destinations in the UP.

Once again, by properly designing the UPs, the TE Fanout Multiplicative Effect can be kept under control, since the path computation is local for each of the UPs.

```
                                  +-------+
           Source                 |  Src  |
                                  | Node  |
                                  +-------+  F1=3
                                 /    |    \
                                /     |     \
                        +-------+ +-------+ +-------+
           Tier 1       |       | |       | |       |
                        |  Node | |  Node | |  Node |
                        +-+-----+ +---+---+ +-----+-+  F2=3
                          |   \ \ /   |   \ / /   |
                          |    \/ \---|---/ \/    |
                          |   / \ /---|---\ / \   |
                          | /  / \    |    / \  \ |
                        +-+-----+ +---+---+ +-----+-+
           Tier 2       |       | |       | |       |
                        |  Node | |  Node | |  Node |
                        +-------+ +-------+ +-------+  F3=2
                          \      \ /     \ /     /
                           \     / \     / \    /
                            \   /    / \    \  /
                             +-------+ +-------+
                             |       | |       |
           Tier 3            | Node  | | Node  |
                             +-------+ +-------+  F4=1
                                 \        /
                                  \      /
                                  +-------+
                                  | Dest  |
           Destination            | Node  |
                                  +-------+
```

Figure 5. Fan out multiplicative effect with TE.

To be completed.

5. HSDN Label Stack Assignment Scheme

HSDN can use any scheme to assign the labels in the label stack. However, if a label assignment scheme which assigns labels in a way consistent with the HSDN structure, important simplifications can be achieved in the control plane and in the LFIBs.

For non-TE FECs, the HSDN label assignment scheme assigns labels according to the "physical" location of the endpoint in the HSDN structure. Continuing our design example from above, for an endpoint X in UP2-a, PL0 would identify the DC in which the endpoint is located, PL1 would identify the group of racks in which the endpoint is located within the DC, and PL2 would identify the endpoint within the group of servers within the DC.

For TE FECs, the HSDN label assignment scheme assigns labels to identify a specific path in each UP that is traversed. In our example, for a specific TE tunnel to endpoint X, PL0 would identify the specific path that should be followed in the DCI, PL1 would identify the path that should be followed within the DC to reach the group of racks, and PL2 would identify the path to reach the endpoint within the group of racks (if there are multiple paths).

In order to assign labels to both non-TE traffic and TE traffic, HSDN uses a label format in which the labels are divided into two logical sub-fields, one identifying the destination within the UP, called Destination Identifier (DID), and one identifying the path, called Path Identifier (PID). The Path Identifier is only relevant for TE traffic, and can be zero for non-TE traffic. The HSDN Label format is illustrated in Figure 6.

```
0                           d                        19
+++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Destination Identifier  |     Path Identifier    |
+++++++++++++++++++++++++++++++++++++++++++++++++++++
|<-------- d bits -------->|<-----(20-d bits)------>|
LSB                                                MSB
```

Figure 6. HSDN Label format.

Depending on the LFIB configuration, the two MSBs may be reserved for identifying the layer (i.e., whether the label is PL0, PL1, or PL2) to resolve ambiguity (not shown in Figure 6).

By properly designing the UPs, this label assignment scheme can support the desired scalability and the support of end-to-end TE traffic.

Note that by using this type of label assignment scheme important benefits can be achieved, including:

- The LFIBs become rather "static," since the FECs are tied to "physical" locations and paths, which change infrequently. This simplifies the use of the SDN approach to configure the LFIBs via a controller.

- All paths in each ECMP group use the same outgoing labels. This guarantees that a single LFIB entry can be used for each ECMP group.

The label stack needs to be imposed at the entry points. For an endpoint, this implies that the server NIC must be able to push a three-label stack of path labels (in addition to possibly one additional VL label for the overlay network).

6. HSDN Architecture - Control Plane

HSDN has been designed to support the controller-centric SDN approach in a scalable fashion. HSDN also supports the traditional distributed control plane approach.

HSDN introduces important simplifications in the control plane and in the network state as well.

6.1. The SDN approach

In the controller-centric SDN approach, the SDN controller configures the LFIBs in all the network nodes. With HSDN, the hierarchical partitioned structure offers a natural framework for a distributed implementation of the SDN controller, since the control plane in each UP is largely independent from other UPs.

For example, a possible architecture uses a SDN controller for each UP. Such SDN partition controller is in charge of configuring the LFIBs in the network nodes in the corresponding UP.

The SDN partition controller may also be in charge of TE computation. With proper design of the UPs, TE path computation algorithms which perform partition-local computation while approach global optimality can be used.

To be completed.

6.2. Distributed control plane

HSDN can also use the traditional distributed routing protocol approach to distribute HSDN labels, for example using BGP [RFC3107].

To be completed.

7. Security Considerations

When the SDN approach is used, the protocols used to configure the LFIBs in the network nodes MUST be mutually authenticated.

For general MPLS/GMPLS security considerations, refer to [RFC5920].

Given the potentially very large scale and the dynamic nature in the cloud/DC environment, the choice of key management mechanisms need to be further studied.

To be completed.

8. IANA Considerations

TBD.

9. References

9.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

9.2 Informative References

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

Authors’ Addresses

Luyuan Fang
Microsoft
5600 148th Ave NE
Redmond, WA 98052
Email: lufang@microsoft.com

Vijay Gill
Microsoft
5600 148th Ave NE
Redmond, WA 98052
Email: vgill@microsoft.com

Fabio Chiussi
Seattle, WA
Email: fabiochiussi@gmail.com

MPLS Working Group
Internet-Draft
Intended Status: Standards Track
Expires: April 30, 2015

Rakesh Gandhi
Tarek Saad
Robert Sawaya
Cisco Systems, Inc.

October 27, 2014

YANG Data Model for MPLS Traffic Engineering Tunnels and Links
draft-gandhi-mpls-te-yang-model-01

Abstract

This document defines YANG data model for the management of Multi-Protocol Label Switching Traffic Engineering (MPLS-TE) tunnels, Label Switched Paths (LSPs) and links. The data model covers configuration data, operational state data, execution commands and event notifications.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines YANG [RFC6020] data model for the management of Multi-Protocol Label Switching Traffic Engineering (MPLS-TE) [RFC3209] tunnels and links. Resource Reservation Protocol (RSVP) [RFC2205] signaled MPLS-TE paths can be represented as tunnels at the head-end Label Switching Router (LSR), and as Label Switched Paths (LSPs) at the head-end, mid-point and tail-end LSRs.

The data model defined in this document includes configuration data, operational state data (status information and counters), execution requests using RPCs (Remote Procedure Calls), and event notifications pertaining to MPLS-TE tunnels, LSPs and MPLS-TE enabled links, as well as system-wide global MPLS-TE properties that relate to the behavior and operation of the TE enabled LSR node.

Further modules augmenting this data model with advanced features can be handled in a future revision or a separate document.

2. Terminology and Notation

2.1. Terminology

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

2.2. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

3. Objectives

This section outlines some of the design objectives for the model:

- In case of existing implementations, it needs to map the data model defined in this document to their proprietary native data model. To facilitate such mappings, the data model should be simple.

- The data model should be suitable for new implementations to use as is, without requiring a mapping to a different native model.

- Mapping to the MPLS-TE MIB Module should be clear.

- The data model should include read-only counters in order to gather statistics for sent and received octets and packets, received packets with errors, and packets that could not be sent due to errors.

- It should be straight forward to augment the base data model for advanced MPLS-TE features.

4. MPLS-TE Data Models Overview

MPLS-TE YANG data models are defined for various management components including configuration, operational state, execution commands and event notifications. Following sections provide overview and some selective examples of these management components for global MPLS-TE, MPLS-TE tunnels, MPLS-TE LSPs and MPLS-TE enabled links.

4.1. Global MPLS-TE Data Model Overview

This module defines YANG data model for configuration data, operational state data, execution commands and event notifications globally for MPLS-TE features.

1. Global MPLS-TE configuration data model: The global MPLS-TE configuration data model is a read-write YANG data model that controls the LSR behavior system-wide. Examples of such configuration items for global MPLS-TE are:

   - Auto-tunnel backup: controls and manages the automatic creation of fast reroute backup tunnels for protected MPLS-TE enabled links.
   - Auto-tunnel mesh-group: controls and manages the automatic creation of tunnels for mesh-groups.
   - Auto-tunnel one-hop: controls and manages the automatic

creation of 1-hop tunnels on MPLS-TE enabled links.

- Auto-bandwidth parameters: controls and manages the

  auto-bandwidth specific system-wide properties.
- System-wide Point-to-Multipoint (P2MP) TE parameters
- Names for SRLG values
- Names for link (extended) administrative groups (AG, EAG)
- MPLS-TE Diff-Serve classes: controls and manages the Diff-Serve TE (DS-TE) model and TE-class maps
- System-wide capabilities for TE LSP re-optimization e.g.
  - Re-optimization times (periodic interval, installation, cleanup)
- System-wide capabilities for MPLS TE link flooding e.g.
  - Periodic flooding interval, bandwidth threshold values.
- MPLS-TE tunnel templates: These are templates that can be used to instantiate tunnels and LSPs with identical configuration properties.
- System-wide capabilities that affect the originating, traversing and terminating LSPs. For example:
  - CSPF metric (TE or IGP)
  - Handling for over-loaded nodes
  - (Soft) preemption parameters
  - Path protection parameters at the head-end LSR
  - Fast reroute parameters

2. Global MPLS-TE state data model: The global MPLS-TE state data model is a read-only YANG data model. This module defines system-wide operational data for various MPLS-TE features. Examples of such system-wide MPLS-TE states are:

   - Global statistics (signaling, admission, preemption, flooding)
   - Global counters (number of tunnels/LSPs/links)

3. Global MPLS-TE execution data model: The global MPLS-TE execution model facilitates issuing commands to an LSR node and optionally returning responses. This model uses RPC operations and contains optional read-only input and output data. Examples of such global MPLS-TE commands are:

   - Clear global MPLS-TE statistics of various features

4. Global MPLS-TE events notification data model: The global MPLS-TE events notification model uses configuration data for registration. Node notifies the registered events to the server using notification messages. Notifications carry read-only data in the messages. Example of such global MPLS-TE events are:

   - Backup tunnel FRR active and not-active state transition events

4.2. MPLS-TE Tunnel Data Model Overview

This module defines configuration data, operational state data, execution commands and event notifications for MPLS-TE tunnels and is applicable to head-end LSRs.

1. MPLS-TE tunnels configuration data model: The configuration data model is a read-write YANG data model. This module defines configuration items for the MPLS-TE tunnels. Examples of such configuration items are:

   - Name
   - Admin-state
   - Tunnel-type (such as P2P, P2MP)
   - Routing properties (such as auto-route announce, forwarding adjacency)
   - Forwarding properties (such as traffic class based tunnel selection)
   - Static route information
   - LDP over tunnel parameters
   - Quality of Service (QoS) policy parameters

2. MPLS-TE tunnel state data model: The MPLS-TE tunnel state data model is a read-only YANG data model. This module defines operational state data for MPLS-TE tunnels at the head-end LSRs. Examples of such MPLS-TE tunnel states are:

   - Name
   - Tunnel creation information (time and trigger: static-configuration/auto-tunnel)
   - State information (Up/Down: when and reason)
   - Traffic counters
   - History of events

3. MPLS-TE tunnel execution data model: The execution model facilitates issuing commands to an LSR node and optionally returning responses. This model uses RPC operations and contains optional read-only input and output data. Example commands for MPLS-TE tunnels are:

   - Clear statistics for all or for individual tunnels

4. MPLS-TE tunnel events notification data model: The notification model uses configuration data for registration. Node notifies the registered events to the server using notification messages. Notifications carry read-only data in the messages. Example events for MPLS-TE tunnels are:

- Tunnel creation and deletion events
- Tunnel state transition events

4.3. MPLS-TE Tunnel LSP Data Model Overview

This module defines configuration data, operational state data, execution commands and event notifications for MPLS-TE tunnel LSPs and is applicable to head-end, mid-point and tail-end LSRs.

1. MPLS-TE tunnel LSP configuration data model: The configuration data model is a read-write YANG data model component. This module defines configuration items for the MPLS-TE tunnel LSP properties. Examples of such MPLS-TE tunnel LSP configuration items are:

   - Name
   - Signaling properties (such as bandwidth, class-type, set-up and hold priorities)
   - Path-computation parameters (dynamic path, explicit path, cost-limit, hop-limit, metric type, affinity parameters (colors))

2. MPLS-TE tunnel LSP state data model: The MPLS-TE tunnel LSP state data model is a read-only YANG data model. This model defines the operational state data for MPLS-TE tunnel LSPs for head-end, mid-point and tail-end LSRs. Example states data for MPLS-TE tunnel LSPs are:

   - Name
   - LSP creation information (time)
   - State information (Up/Down: when and reason)
   - MPLS-TE attribute-set name
   - Signaling information (Explicit Route Object, Record Route Object, bandwidth, egress and ingress links)
   - FRR information (status, type of protection, backup tunnel)
   - Soft preemption properties
   - Path protection properties
   - Statistics
   - History of events

3. MPLS-TE tunnel LSP execution data model: The execution model facilitates issuing commands to an LSR node and optionally returning responses. This model uses RPC operations and contains optional read-only input and output data. Examples of such commands for MPLS-TE tunnel LSPs are:

   - Trigger re-optimization on all or on individual LSP
   - Trigger path protection switchover on an individual LSP
   - Trigger LSP path switchover on an individual LSP

- Clear TE statistics for all or for individual LSPs

4. MPLS-TE tunnel LSP events notification data model: The notification model uses configuration data for registration. Node notifies the registered events to the server using notification messages. Notifications carry read-only data in the messages. Examples of such events for MPLS-TE LSPs are:

   - LSP creation and deletion events
   - LSP state transition events
   - LSP re-optimization including trigger reason
   - Fast Reroute (protection availability, activation) events
   - LSP signaling events
   - Auto-bandwidth changes
   - Path protection events
   - (Soft) Preemption events

4.4. MPLS-TE Link Data Model Overview

This module defines configuration data, operational state data, execution commands and event notifications for MPLS-TE enabled links on an LSR.

1. MPLS-TE link configuration data model: The configuration data model is a read-write YANG data model component. This model defines configuration items for MPLS-TE enabled links used to advertise in TE topology database. Examples of such configuration items for MPLS-TE enabled links are:

   - Name
   - Maximum reservable bandwidth, bandwidth constraints (BC) values
   - Diff-serve model type (e.g. Russian Doll Model)
   - (Extended) Administrative groups (AGs and EAGs)
   - SRLG values
   - TE metric value
   - Flooding parameters
     - Flooding intervals and threshold values
   - RSVP Parameters
     - RSVP authentication parameters
     - RSVP refresh reduction parameters
     - RSVP hello parameters
     - RSVP graceful restart (GR)
   - Fast reroute backup tunnel properties (such as static, auto-tunnel)

2. MPLS-TE link state data model: The MPLS-TE link state model is a read-only YANG data model. This model defines operational state data

for MPLS-TE enabled links for an LSR node. Examples of such state data for MPLS-TE links are:

- Name
- State information (UP/Down: when and reason)
- IGP information
    - IGP neighbor
    - IGP metric
- Bandwidth information: maximum bandwidth, reserved bandwidth at different priorities, available bandwidth at different priorities and for each class-type (CT)
- List of admitted LSPs (name, bandwidth value and pool, time, priority)
- Statistics: state counters, flooding counters, admission counters (accepted/rejected), preemption counters
- History of events

3. MPLS-TE link execution data model: The execution model facilitates issuing commands to an LSR node and optionally returning responses. This model uses RPC operations and contains optional read-only input and output data. Examples of such commands for MPLS-TE links are:

- Clear TE statistics for all or for individual links
- Trigger immediate flooding for all TE links

4. MPLS-TE link events notification data model: The notification model uses configuration data for registration. Node notifies the registered events to the server using notification messages. Notifications carry read-only data in the messages. Example events for MPLS-TE links are as following:

- Link creation and deletion events
- Link state transition events
- (Soft) preemption trigger events
- Fast reroute activation events

5. High-level Tree Structure of MPLS-TE Data Model

The module, "ietf-mpls-te", defines the YANG data model for various management components (configuration, operational, execution and notification) within MPLS-TE. The data module includes modules for global MPLS-TE, tunnels, LSPs and links and the tree structure is organized as shown below.

The following notations are used for the data tree.

```
   <status> <flags> <name> <opts> <type>

    <status> is one of:

      +  for current
      x  for deprecated
      o  for obsolete

    <flags> is one of:
      rw  for read-write configuration data
      ro  for read-only non-configuration data
      -x  for execution rpcs
      -n  for notifications

    <name> is the name of the node

    If the node is augmented into the tree from another module, its name
   is printed as <prefix>:<name>

    <opts> is one of:
      ? for an optional leaf or node
      ! for a presence container
      * for a leaf-list or list
      Brackets [<keys>] for a list’s keys
      Curly braces {<condition>} for optional feature that make node
   conditional
      Colon : for marking case nodes
      Ellipses ("...") subtree contents not shown

    <type> is the name of the type for leafs and leaf-lists.


   module ietf-mpls-te
       +--rw tunnels-cfg!
       +--rw lsps-cfg!
       +--rw links-cfg!
       +--rw global-cfg!
       +--ro tunnels-oper
       +--ro lsps-oper
       +--ro links-oper
       +--ro global-oper
    rpcs:
       +---x tunnels-rpc
       +---x lsps-rpc
       +---x global-rpc
       +---x links-rpc
    notifications:
       +---n tunnels-notif
       +---n lsps-notif
       +---n links-notif
       +---n global-notif
```

As shown, data tree structure is organized by MPLS-TE data modules, which are global MPLS-TE, tunnels, LSPs and links.
Each of this data model module contains various management components including configuration, operation, execution and notification.

6. MPLS-TE Global Data Model Subtree Structure

This document defines the YANG data model subtree for global MPLS TE configuration, state, RPCs and notifications as follows:

```
   +--rw global-cfg!
   |  +--rw path-selection
   |  |  +--rw cost-limit?   uint32
   |  |  +--rw hop-limit?    uint32
   |  |  +--rw metric?       mtt:path-metric-type
   |  |  +--rw tiebreaker?   mtt:path-tiebreaker-type
   |  +--rw bfd!
   |  |  +--rw type?               mtt:bfd-type
   |  |  +--rw encap-mode?         mtt:bfd-encap-mode-type
   |  |  +--rw bringup-timeout?    uint32
   |  |  +--rw dampening?          uint32
   |  |  +--rw lsp-ping
   |  |  |  +--rw disable?    boolean
   |  |  |  +--rw interval?   uint32
   |  |  +--rw minimum-interval?   uint32
   |  |  +--rw multiplier?         uint32
   |  +--rw logging-events
   |  |  +--rw (type)?
   |  |     +--:(all)
   |  |     |  +--rw all!
   |  |     +--:(bfd-status)
   |  |     |  +--rw bfd-status!
   |  |     +--:(link-status)
   |  |     |  +--rw link-status!
   |  |     +--:(lsp-status)
   |  |     |  +--rw lsp-status!
   |  |     |     +--rw all?      boolean
   |  |     |     +--rw events*   mtt:lsp-status-event-type
   |  |     +--:(cspf-failure)
   |  |        +--rw cspf-failure!
   |  +--rw tunnel-templates* [name]
   |  |  +--rw name                     string
   |  |  +--rw source?                  inet:ip-address
   |  |  +--rw fast-reroute!
   |  |  |  +--rw bandwidth-protect?   boolean
   |  |  |  +--rw node-protect?        boolean
   |  |  +--rw record-route?           boolean
   |  |  +--rw signaled-name?          string
   |  |  +--rw priority
   |  |  |  +--rw setup?   uint8
   |  |  |  +--rw hold?    uint8
   |  |  +--rw soft-preemption?        boolean
   |  |  +--rw signaled-bandwidth
   |  |  |  +--rw type?    mtt:bandwidth-type
   |  |  |  +--rw value?   uint32
   |  |  +--rw (style)?
   |  |  |  +--:(affinity-hex)
   |  |  |  |  +--rw affinity-hex
   |  |  |  |     +--rw value?   uint32
   |  |  |  |     +--rw mask?    uint32
   |  |  |  +--:(affinity-name)
   |  |  |     +--rw affinity-name
   |  |  |        +--rw constraints* [action]
   |  |  |           +--rw action        mtt:affinity-action-type
   |  |  |           +--rw constraint
   |  |  |              +--rw affinity-list* [name]
   |  |  |                 +--rw name    string
   |  |  +--rw logging-events
   |  |  |  +--rw (type)?
   |  |  |     +--:(all)
   |  |  |     |  +--rw all!
   |  |  |     +--:(bfd-status)
   |  |  |     |  +--rw bfd-status!
   |  |  |     +--:(link-status)
   |  |  |     |  +--rw link-status!
   |  |  |     +--:(lsp-status)
   |  |  |     |  +--rw lsp-status!
   |  |  |     |     +--rw all?      boolean
   |  |  |     |     +--rw events*   mtt:lsp-status-event-type
   |  |  |     +--:(cspf-failure)
   |  |  |        +--rw cspf-failure!
   |  |  +--rw auto-bandwidth!
   |  |  |  +--rw overflow-threshold?      uint32
   |  |  |  +--rw overflow-limit?          uint8
   |  |  |  +--rw underflow-threshold?     uint32
   |  |  |  +--rw underflow-limit?         uint8
   |  |  |  +--rw collect-only?            boolean
   |  |  |  +--rw application-frequency?   uint32
   |  |  |  +--rw bandwidth-limit
   |  |  |     +--rw min-limit?   uint32
   |  |  |     +--rw max-limit?   uint32
   |  |  +--rw (routing-proprties)?
   |  |  |  +--:(autoroute)
   |  |  |  |  +--rw autoroute!
   |  |  |  |     +--rw routing-afs*       inet:ip-version
   |  |  |  |     +--rw (metric-type)?
   |  |  |  |        +--:(metric)
   |  |  |  |        |  +--rw metric?            uint32
   |  |  |  |        +--:(relative-metric)
   |  |  |  |        |  +--rw relative-metric?   uint32
   |  |  |  |        +--:(absolute-metric)
   |  |  |  |           +--rw absolute-metric?   uint32
   |  |  |  +--:(forwarding-adjacency)
   |  |  |     +--rw forwarding-adjacency!
   |  |  |        +--rw holdtime?      uint32
   |  |  |        +--rw routing-afs*   inet:ip-version
   |  |  +--rw (forwarding-property)?
   |  |     +--:(forwarding-class)
   |  |     |  +--rw forwarding-class
   |  |     |     +--rw class?   uint8
   |  |     +--:(forwarding-group)
   |  |        +--rw forwarding-group
   |  |           +--rw classes*   uint8
   |  +--rw auto-tunnel-mesh* [mesh-group]
   |  |  +--rw mesh-group       uint32
   |  |  +--rw template-name?   string
   |  |  +--rw one-hop-only?    boolean
   |  +--rw template-name?           string
   |  +--rw nhop-only?               boolean
   |  +--rw load-share-properties
   |  |  +--rw unequal!
   |  |     +--rw bandwidth-based?   boolean
   |  +--rw global-timers
   |  |  +--rw lsp-bw-hold-delay?          uint32
   |  |  +--rw lsp-preemption-delay?       uint32
   |  |  +--rw topology-holddown-sigerr?   uint32
   |  +--rw topology-flooding!
   |  |  +--rw flooding-interval?     uint32
   |  |  +--rw flooding-thresholds
   |  |     +--rw (type)?
   |  |        +--:(single-step)
   |  |        |  +--rw up-step?      uint8
   |  |        |  +--rw down-step?    uint8
   |  |        +--:(multiple-steps)
   |  |           +--rw up-steps* [value]
   |  |           |  +--rw value    uint8
   |  |           +--rw down-steps* [value]
   |  |              +--rw value    uint8
   |  +--rw global-reoptimization!
   |  |  +--rw interval?             uint32
   |  |  +--rw installation-delay?   uint32
   |  |  +--rw cleanup-delay?        uint32
   |  +--rw softpreemption-te!
   |  |  +--rw softpremption-timeout?   uint32
   |  +--rw affinity-maps
   |  |  +--rw affinity-maps* [name]
   |  |     +--rw name         string
   |  |     +--rw (type)?
   |  |        +--:(value)
   |  |        |  +--rw value?       uint32
   |  |        +--:(bit-index)
   |  |           +--rw bit-index?   uint32
   |  +--rw srlg-maps
   |  |  +--rw srlg-maps* [name]
   |  |     +--rw name            string
   |  |     +--rw value?          uint32
   |  |     +--rw admin-weight?   uint32
   |  +--rw ds-te!
   |  |  +--rw bc-model?   mtt:bc-model-type
   |  |  +--rw te-class* [class-index]
   |  |     +--rw class-index    uint8
   |  |     +--rw priority?      uint8
   |  |     +--rw bc-value?      uint8
   |  +--rw auto-bw!
   |  |  +--rw stats-collection-interval?   uint32
   |  +--rw tail-signaling
   |  |  +--rw advertise-explicit-null?   boolean

   +--ro global-state
   |   +--ro tunnel-templates-state
   |   |  +--ro tunnel-template*  [name]
   |   |  +--ro name                    string

   +---x global-rpcs
   |   +---x te-global-rpcs-to-be-defined

   +---n global-notifications
   |   +---n te-global-notifications-to-be-defined
```

6.1. MPLS-TE Global Tunnel-template Lists

The data model for tunnel-templates presented in this document uses a flat list of tunnel-template(s). Each tunnel-template in the list is identified by its name. A tunnel-template is a configuration template that can be used to instantiate tunnels and LSPs with identical properties.

There is one list for MPLS-TE tunnel-template configurations ("/global-cfg/tunnel-templates") and a separate list for operational state data ("/global-state/tunnel-templates").

7. MPLS-TE Tunnel Data Model Subtree Structure

This document defines the YANG data model subtree for TE tunnel configuration, state, RPCs and notifications as follows:

```
   +--rw tunnels-cfg!
   |  +--rw tunnel* [name type]
   |     +--rw name                    string
   |     +--rw type                    mtt:tunnel-type
   |     +--rw identifier?             uint16
   |     +--rw description?            string
   |     +--rw admin-mode?             enumeration
   |     +--rw path-protection?        boolean
   |     +--rw backup?                 boolean
   |     +--rw load-share?             uint32
   |     +--rw auto-bandwidth!
   |     |  +--rw overflow-threshold?      uint32
   |     |  +--rw overflow-limit?          uint8
   |     |  +--rw underflow-threshold?     uint32
   |     |  +--rw underflow-limit?         uint8
   |     |  +--rw collect-only?            boolean
   |     |  +--rw application-frequency?   uint32
   |     |  +--rw bandwidth-limit
   |     |     +--rw min-limit?   uint32
   |     |     +--rw max-limit?   uint32
   |     +--rw (forwarding-property)?
   |     |  +--:(forwarding-class)
   |     |  |  +--rw forwarding-class
   |     |  |     +--rw class?   uint8
   |     |  +--:(forwarding-group)
   |     |     +--rw forwarding-group
   |     |        +--rw classes*   uint8
   |     +--rw (routing-proprties)?
   |     |  +--:(autoroute)
   |     |  |  +--rw autoroute!
   |     |  |     +--rw routing-afs*       inet:ip-version
   |     |  |     +--rw (metric-type)?
   |     |  |        +--:(metric)
   |     |  |        |  +--rw metric?            uint32
   |     |  |        +--:(relative-metric)
   |     |  |        |  +--rw relative-metric?   uint32
   |     |  |        +--:(absolute-metric)
   |     |  |           +--rw absolute-metric?   uint32
   |     |  +--:(forwarding-adjacency)
   |     |     +--rw forwarding-adjacency!
   |     |        +--rw holdtime?      uint32
   |     |        +--rw routing-afs*   inet:ip-version
   |     +--rw backup-bandwidth
   |     |  +--rw value?   uint32
   |     |  +--rw type?    mtt:backup-bandwidth-type
   |     +--rw bidirectional
   |     |  +--rw association
   |     |     +--rw id?              uint32
   |     |     +--rw source?          inet:ip-address
   |     |     +--rw global-source?   inet:ip-address
   |     |     +--rw type?            mtt:bidir-association-type
   |     +--rw destination* [address]
   |     |  +--rw address    inet:ip-address
   |     |  +--rw paths* [index]
   |     |     +--rw index             uint8
   |     |     +--rw (type)?
   |     |     |  +--:(dynamic)
   |     |     |  |  +--rw dynamic?          boolean
   |     |     |  +--:(explicit)
   |     |     |     +--rw explicit
   |     |     |        +--rw hops* [index]
   |     |     |           +--rw index          uint8
   |     |     |           +--rw hop-address?   mtt:hop-address-type
   |     |     |           +--rw hop-action?    mtt:hop-action-type
   |     |     +--rw igp-constraint
   |     |     |  +--rw igp?          enumeration
   |     |     |  +--rw area-level?   uint32
   |     |     +--rw no-cspf?          boolean
   |     |     +--rw lockdown?         boolean
   |     +--rw (tunnel-type)?
   |        +--:(p2p-properties)
   |        |  +--rw p2p-properties* [index]
   |        |     +--rw destination?          leafref
   |        |     +--rw index                 leafref
   |        |     +--rw source?               inet:ip-address
   |        |     +--rw fast-reroute!
   |        |     |  +--rw bandwidth-protect?   boolean
   |        |     |  +--rw node-protect?        boolean
   |        |     +--rw record-route?         boolean
   |        |     +--rw signaled-name?        string
   |        |     +--rw priority
   |        |     |  +--rw setup?   uint8
   |        |     |  +--rw hold?    uint8
   |        |     +--rw soft-preemption?      boolean
   |        |     +--rw signaled-bandwidth
   |        |     |  +--rw type?    mtt:bandwidth-type
   |        |     |  +--rw value?   uint32
   |        |     +--rw (style)?
   |        |     |  +--:(affinity-hex)
   |        |     |  |  +--rw affinity-hex
   |        |     |  |     +--rw value?   uint32
   |        |     |  |     +--rw mask?    uint32
   |        |     |  +--:(affinity-name)
   |        |     |     +--rw affinity-name
   |        |     |        +--rw constraints* [action]
   |        |     |           +--rw action   mtt:affinity-action-type
   |        |     |           +--rw constraint
   |        |     |              +--rw affinity-list* [name]
   |        |     |                 +--rw name    string
   |        |     +--rw path-selection
   |        |     |  +--rw cost-limit?   uint32
   |        |     |  +--rw hop-limit?    uint32
   |        |     |  +--rw metric?       mtt:path-metric-type
   |        |     |  +--rw tiebreaker?   mtt:path-tiebreaker-type
   |        |     +--rw bfd!
   |        |     |  +--rw type?               mtt:bfd-type
   |        |     |  +--rw encap-mode?         mtt:bfd-encap-mode-type
   |        |     |  +--rw bringup-timeout?    uint32
   |        |     |  +--rw dampening?          uint32
   |        |     |  +--rw lsp-ping
   |        |     |  |  +--rw disable?    boolean
   |        |     |  |  +--rw interval?   uint32
   |        |     |  +--rw minimum-interval?   uint32
   |        |     |  +--rw multiplier?         uint32
   |        |     +--rw logging-events
   |        |        +--rw (type)?
   |        |           +--:(all)
   |        |           |  +--rw all!
   |        |           +--:(bfd-status)
   |        |           |  +--rw bfd-status!
   |        |           +--:(link-status)
   |        |           |  +--rw link-status!
   |        |           +--:(lsp-status)
   |        |           |  +--rw lsp-status!
   |        |           |     +--rw all?      boolean
   |        |           |     +--rw events*   mtt:lsp-status-event-type
   |        |           +--:(cspf-failure)
   |        |              +--rw cspf-failure!
   |        +--:(p2mp-properties)
   |           +--rw p2mp-properties* [lsp-index]
   |              +--rw lsp-index            uint32
   |              +--rw p2mp-path-group* [path-index]
   |              |  +--rw path-index     uint32
   |              |  +--rw destination?   leafref
   |              |  +--rw index?         leafref
   |              +--rw source?             inet:ip-address
   |              +--rw fast-reroute!
   |              |  +--rw bandwidth-protect?   boolean
   |              |  +--rw node-protect?        boolean
   |              +--rw record-route?         boolean
   |              +--rw signaled-name?        string
   |              +--rw priority
   |              |  +--rw setup?   uint8
   |              |  +--rw hold?    uint8
   |              +--rw soft-preemption?      boolean
   |              +--rw signaled-bandwidth
   |              |  +--rw type?    mtt:bandwidth-type
   |              |  +--rw value?   uint32
   |              +--rw (style)?
   |              |  +--:(affinity-hex)
   |              |  |  +--rw affinity-hex
   |              |  |     +--rw value?   uint32
   |              |  |     +--rw mask?    uint32
   |              |  +--:(affinity-name)
   |              |     +--rw affinity-name
   |              |        +--rw constraints* [action]
   |              |           +--rw action   mtt:affinity-action-type
   |              |           +--rw constraint
   |              |              +--rw affinity-list* [name]
   |              |                 +--rw name    string
   |              +--rw path-selection
   |              |  +--rw cost-limit?   uint32
   |              |  +--rw hop-limit?    uint32
   |              |  +--rw metric?       mtt:path-metric-type
   |              |  +--rw tiebreaker?   mtt:path-tiebreaker-type
   |              +--rw bfd!
   |              |  +--rw type?              mtt:bfd-type
   |              |  +--rw encap-mode?        mtt:bfd-encap-mode-type
   |              |  +--rw bringup-timeout?    uint32
   |              |  +--rw dampening?          uint32
   |              |  +--rw lsp-ping
   |              |  |  +--rw disable?    boolean
   |              |  |  +--rw interval?   uint32
   |              |  +--rw minimum-interval?   uint32
   |              |  +--rw multiplier?         uint32
   |              +--rw logging-events
   |                 +--rw (type)?
   |                    +--:(all)
   |                    |  +--rw all!
   |                     +--:(bfd-status)
   |                     |  +--rw bfd-status!
   |                     +--:(link-status)
   |                     |  +--rw link-status!
   |                     +--:(lsp-status)
   |                     |  +--rw lsp-status!
   |                     |     +--rw all?      boolean
   |                     |     +--rw events*   mtt:lsp-status-event-type
   |                     +--:(cspf-failure)
   |                        +--rw cspf-failure!


   +---x tunnels-state
   |  +---x tunnels-state-to-be-defined

   +---x tunnels-rpc
   |  +---x tunnels-rpcs-to-be-defined

   +---n tunnels-notif
   |  +---n tunnels-notifications-to-be-defined
```

7.1. MPLS-TE Tunnel Lists

The data model for tunnels-cfg presented in this document uses a flat list of tunnels. Each tunnel in the list is identified by its name and a mandatory "tunnel-type".

There is one list for configured MPLS-TE tunnels ("/tunnels-cfg/tunnel"), and a separate list for the operational state of all MPLS-TE tunnels ("/tunnels-state/tunnel").

8. MPLS-TE Tunnel LSPs Data Model Subtree Structure

This document defines the YANG data model subtree for TE LSP configuration, state, RPCs and notifications as

the following tree structure:

```
      +--rw lsps-cfg
      |  +--rw lsp* [name]
      |  |  +--rw name               string
      |  |  +--rw lsp-configuration-to-be-defined
      +--ro lsps-state
      |  +--ro lsp* [name]
      |  |  +--ro name               string
      |  |  +--ro lsp-operational-state-to-be-defined

      +---x lsps-rpc
      |  +---x lsp-rpcs-to-be-defined

      +---n lsps-notif
      |  +---n lsp-notifications-to-be-defined
```

8.1. MPLS-TE Tunnel LSP Lists

The data model for lsps-cfg presented in this document uses a flat list of lsps. Each LSP in the list is identified by its name.

There is one list for MPLS-TE tunnel LSP configurations ("/lsps-cfg/lsp"), and a separate list for the operational state of all MPLS-TE tunnel LSPs ("/lsps-state/lsp").

9. MPLS-TE Links Data Model Subtree Structure

This document defines the YANG data model subtree for TE link configuration, state, RPCs and notifications as follows:

```
   +--rw links-cfg
   |  +--rw link* [name]
   |     +--rw name                      string
   |     +--rw admin-weight?             uint32
   |     +--rw reservable-bandwidths
   |     |  +--rw bc-model?                mtt:bc-model-type
   |     |  +--rw value-type?              boolean
   |     |  +--rw bandwidth-constraints
   |     |     +--rw maximum-reservable?   uint32
   |     |     +--rw bc0?                  uint32
   |     |     +--rw bc1?                  uint32
   |     |     +--rw bc2?                  uint32
   |     |     +--rw bc3?                  uint32
   |     |     +--rw bc4?                  uint32
   |     |     +--rw bc5?                  uint32
   |     |     +--rw bc6?                  uint32
   |     |     +--rw bc7?                  uint32
   |     +--rw flooding-thresholds
   |     |  +--rw (type)?
   |     |     +--:(single-step)
   |     |     |  +--rw up-step?      uint8
   |     |     |  +--rw down-step?    uint8
   |     |     +--:(multiple-steps)
   |     |        +--rw up-steps* [value]
   |     |        |  +--rw value    uint8
   |     |        +--rw down-steps* [value]
   |     |           +--rw value    uint8
   |     +--rw link-affinities
   |     |  +--rw (type)?
   |     |     +--:(bitmap)
   |     |     |  +--rw bitmap?            uint32
   |     |     +--:(bitmap-extended)
   |     |     |  +--rw bitmap-extended?   uint32
   |     |     +--:(names)
   |     |        +--rw names* [name]
   |     |           +--rw name    string
   |     +--rw shared-risk-link-group* [name]
   |     |  +--rw name    string
   |     +--rw fast-reroute-backup
   |        +--rw (type)?
   |           +--:(configured-backups)
   |           |  +--rw configured-backups* [name]
   |           |     +--rw name    string
   |           +--:(auto-backup)
   |              +--rw name?                      string
   |              +--rw source?                    inet:ip-address
   |              +--rw fast-reroute!
   |              |  +--rw bandwidth-protect?   boolean
   |              |  +--rw node-protect?        boolean
   |              +--rw record-route?              boolean
   |              +--rw signaled-name?             string
   |              +--rw priority
   |              |  +--rw setup?   uint8
   |              |  +--rw hold?    uint8
   |              +--rw soft-preemption?           boolean
   |              +--rw signaled-bandwidth
   |              |  +--rw type?    mtt:bandwidth-type
   |              |  +--rw value?   uint32
   |              +--rw (style)?
   |              |  +--:(affinity-hex)
   |              |  |  +--rw affinity-hex
   |              |  |     +--rw value?   uint32
   |              |  |     +--rw mask?    uint32
   |              |  +--:(affinity-name)
   |              |     +--rw affinity-name
   |              |        +--rw constraints* [action]
   |              |           +--rw action    mtt:affinity-action-type
   |              |           +--rw constraint
   |              |              +--rw affinity-list* [name]
   |              |                 +--rw name    string
   |              +--rw logging-events
   |              |  +--rw (type)?
   |              |     +--:(all)
   |              |     |  +--rw all!
   |              |     +--:(bfd-status)
   |              |     |  +--rw bfd-status!
   |              |     +--:(link-status)
   |              |     |  +--rw link-status!
   |              |     +--:(lsp-status)
   |              |     |  +--rw lsp-status!
   |              |     |     +--rw all?      boolean
   |              |     |     +--rw events*   mtt:lsp-status-event-type
   |              |     +--:(cspf-failure)
   |              |        +--rw cspf-failure!
   |              +--rw auto-bandwidth!
   |              |  +--rw overflow-threshold?      uint32
   |              |  +--rw overflow-limit?          uint8
   |              |  +--rw underflow-threshold?     uint32
   |              |  +--rw underflow-limit?         uint8
   |              |  +--rw collect-only?            boolean
   |              |  +--rw application-frequency?   uint32
   |              |  +--rw bandwidth-limit
   |              |     +--rw min-limit?   uint32
   |              |     +--rw max-limit?   uint32
   |              +--rw (routing-proprties)?
   |              |  +--:(autoroute)
   |              |  |  +--rw autoroute!
   |              |  |     +--rw routing-afs*       inet:ip-version
   |              |  |     +--rw (metric-type)?
   |              |  |        +--:(metric)
   |              |  |        |  +--rw metric?            uint32
   |              |  |        +--:(relative-metric)
   |              |  |        |  +--rw relative-metric?   uint32
   |              |  |        +--:(absolute-metric)
   |              |  |           +--rw absolute-metric?   uint32
   |              |  +--:(forwarding-adjacency)
   |              |     +--rw forwarding-adjacency!
   |              |        +--rw holdtime?      uint32
   |              |        +--rw routing-afs*   inet:ip-version
   |              +--rw (forwarding-property)?
   |              |  +--:(forwarding-class)
   |              |  |  +--rw forwarding-class
   |              |  |     +--rw class?   uint8
   |              |  +--:(forwarding-group)
   |              |     +--rw forwarding-group
   |              |        +--rw classes*   uint8
   |              +--rw link-protection-only?      boolean
   |              +--rw bandwidth-protection?      boolean
   |              +--rw backup-path-computation?
                auto-backup-path-computation-type

   +--ro links-state
   |  +--ro link* [name]
   |  |  +--ro name               string
   |  |  +--ro link-operational-state-to-be-defined

   +---x links-rpc
   |  +---x link-rpcs-to-be-defined

   +---n links-notif
   |  +---n link-notifications-to-be-defined
```

9.1. MPLS-TE Link Lists

The data model for links-cfg presented in this document uses a flat list of links. Each link in the list is identified by its name.

There is one list for MPLS-TE link configurations ("/links-cfg/link"), and a separate list for the operational state of all MPLS-TE links ("/links-state/link").

10. MPLS-TE YANG Generic Types Module

```
module ietf-mpls-te-types {

    namespace "urn:cisco:params:xml:ns:yang:mpls-te-types";

    /* Replace with IANA when assigned */
    prefix "mpls-te-types";

    import ietf-inet-types { prefix inet; }

    organization
      "Cisco Systems
       170 West Tasman Drive
       San Jose, CA 95134-1706
       USA";

    contact
```

```
      "Rakesh Gandhi rgandhi@cisco.com
       Tarek Saad tsaad@cisco.com
       Robert Sawaya rsawaya@cisco.com";

    description
        "This module contains a collection of generally
         useful MPLS-TE specific
         derived YANG data types.";

    revision 2014-10-27 {
        description
            "Initial revision.";
    }

    /* Typedefs for MPLS-TE */

    typedef bc-model-type {
        description
           "Diff-Serve TE Bandwidth Constraint model type.";
        type enumeration {
            enum rdm {
                description
                "Russian Doll bandwidth constraint model type.";
            }
            enum mam {
                description
                "Maximum Allocation bandwidth constraint model type.";
            }
            enum mar {
                description
                "Maximum Allocation with Reservation
                bandwidth constraint model type.";
            }
        }
        default rdm;
    }

    typedef bandwidth-type {
        description "MPLS-TE tunnel bandwidth type";
        type enumeration {
            enum CT0;
            enum CT1;
            enum CT2;
            enum CT3;
            enum CT4;
            enum CT5;
            enum CT6;
            enum CT7;
        }
        default CT0;
    }

    typedef lsp-status-event-type {
        description "Tunnel LSP status event type.";
        type enumeration {
            enum bandwidth-change;
            enum insufficient-bandwidth;
            enum record-route;
            enum reroute;
            enum state;
            enum switchover;
        }
    }

    typedef bandwidth-unit-type {
        description "Bandwidth unit type.";
        type enumeration {
            enum Gbps;
            enum Mbps;
            enum Kbps;
        }
        default Kbps;
    }

    typedef backup-bandwidth-type {
        description "FRR backup tunnel bandwidth protection type.";
        type enumeration {
            enum BC0;
            enum BC1;
            enum BC2;
            enum BC3;
            enum BC4;
            enum BC5;
            enum BC6;
            enum BC7;
            enum BC-any;
        }
        default BC-any;
    }

    typedef tunnel-type {
        type enumeration {
            enum p2p {
                description
                    "MPLS-TE point-to-point tunnel type.";
            }
            enum p2mp {
                description
                    "MPLS-TE point-to-multipoint tunnel type.";
            }
        }
        default p2p;
        description
            "Possible MPLS-TE tunnel types, default is point-to-point.";
    }

    typedef hop-address-type {
        type inet:ip-address;
    }

    typedef hop-action-type {
        type enumeration {
            enum include-strict {
                description "Include strict hop.";
            }
            enum include-loose {
                description "Include loose hop.";
            }
            enum exclude {
                description "Exclude strict hop.";
            }
        }
        default "include-strict";
    }

    typedef bfd-type {
        type enumeration {
            enum classical {
                description "BFD classical session type.";
            }
            enum seamless {
                description "BFD seamless session type.";
            }
        }
        default "classical";
    }

    typedef path-metric-type {
        type enumeration {
            enum igp;
            enum te;
        }
        default igp;
        description "Path metric for CSPF.";
    }

    typedef path-tiebreaker-type {
        type enumeration {
            enum min-fill;
            enum max-fill;
            enum random;
        }
        default min-fill;
        description
            "Possible CSPF path tiebreakers for MPLS-TE tunnels.";
    }

    typedef bidir-association-type {
        type enumeration {
            enum corouted;
            enum non-corouted;
        }
        default non-corouted;
        description
            "Possible types of bidirectional tunnel association.";
    }

    typedef bfd-encap-mode-type {
        type enumeration {
            enum gal;
            enum ip;
        }
        default ip;
        description
            "Possible BFD transport modes when running over MPLS-TE
             LSPs.";
    }

        typedef affinity-action-type {
            type enumeration {
                enum include;
                enum exclude;
                enum include-strict;
                enum exclude-all;
            }
            description
                "Possible handling for affinity.";
        }
    }
```

11. MPLS TE Yang Module

```
    module ietf-mpls-te {

    namespace "urn:cisco:params:xml:ns:yang:ietf-mpls-te";

    /* Replace with IANA when assigned */
    prefix "mpls-te";

    import ietf-inet-types { prefix inet; }
    import ietf-mpls-te-types { prefix mtt; }

    organization
      "Cisco Systems
       170 West Tasman Drive
       San Jose, CA 95134-1706
       USA";

    contact
      "Rakesh Gandhi rgandhi@cisco.com
       Tarek Saad tsaad@cisco.com
       Robert Sawaya rsawaya@cisco.com";

    description
        "Data model for MPLS-TE configuration, state, execution and
         notifications.";

    revision 2014-10-27 {
        description
            "Initial revision.";
    }

    /* Groupings for MPLS-TE */
    grouping lsp-properties {

        choice tunnel-type {
            /* Point-to-point LSP properties */
            list p2p-properties {
                when "/tunnels-cfg/tunnel/type = p2p";
                key "index";
                uses path-option-reference;
                    description "An index identifying a set of LSP
                                 properties";
                uses signaling-properties;
                uses affinity-properties;
                uses path-computation-properties;
                uses bfd-properties;
                uses logging-events;
            }

            /* Point-to-multipoint LSP properties */
            list p2mp-properties {
                when "/tunnels-cfg/tunnel/type = p2mp";
                key "lsp-index";
                leaf lsp-index {
                    description "An index identifying a set of LSP
                                 properties";
                    type uint32;
                }
                list p2mp-path-group {
                    key "path-index";
                    description "Index pointing to the configured
                                path as specified in
                                the path-option-reference";
                    leaf path-index {
                        type uint32;
                    }
                    /* definition of the destination
                      and associated path  */
                    uses path-option-reference;
                }
                uses signaling-properties;
                uses affinity-properties;
                uses path-computation-properties;
                uses bfd-properties;
                uses logging-events;
            }
        }
    }

    grouping path-option-reference {
        leaf destination {
            description "A reference to an MPLS-TE tunnel destination";
            type leafref {
                path "/tunnels-cfg/tunnel/destination/address";
            }
        }
        leaf index {
            description "A reference to an MPLS-TE tunnel path-option";
            type leafref {
                path "/tunnels-cfg/tunnel/destination[address=current()
                /../destination]/paths/index";
            }
        }
    }
    grouping path-properties {
        choice type {
            leaf dynamic {
                description
                    "A CSPF dynamically computed path";
                type boolean;
            }
            container explicit {
                description
                    "An operator specified explicit path";
                list hops {
                    key "index";
                    leaf index {
                        type uint8;
                    }
                    leaf hop-address {
                        description
                            "An IP hop address";
                        type mtt:hop-address-type;
                    }
                    leaf hop-action {
                        description
                            "An IP hop action.";
                        type mtt:hop-action-type;
                    }
                }
            }
        }
        container igp-constraint {
            leaf igp {
                description
                    "Constrains the computed path to a specific IGP.";
                type enumeration {
                    enum ospf;
                    enum isis;
                }
            }
            leaf area-level {
                description
                    "Constrains the computed path to a specific IGP
                     area or level";
                type uint32;
            }
        }
        leaf no-cspf {
            description
                "Indicates no validation checks to be attempted on this
                 path";
            type boolean;
        }
        leaf lockdown {
            description
                "Indicates no reoptimization to be attempted for this
                 path.";
            type boolean;
        }
    }

    grouping bfd-properties {
        container bfd {
            presence "Enables BFD fast-detect on the tunnel.";
            leaf type {
                type mtt:bfd-type;
            }
            leaf encap-mode {
                type mtt:bfd-encap-mode-type;
            }
            leaf bringup-timeout {
                type uint32;
            }
            leaf dampening {
                type uint32;
            }
            container lsp-ping {
                leaf disable {
                    type boolean;
                    default false;
                }
                leaf interval {
                    type uint32;
                }
            }
            leaf minimum-interval {
                type uint32;
            }
            leaf multiplier {
                type uint32;
            }
        }
    }

    grouping path-computation-properties {
        container path-selection {
            leaf cost-limit {
                description
                    "The MPLS-TE tunnel path cost limit.";
                type uint32;
            }
            leaf hop-limit {
                description
                    "The MPLS-TE tunnel path hop limit.";
                type uint32;
            }
            leaf metric {
                description
                    "The MPLS-TE tunnel path metric type.";
                type mtt:path-metric-type;
            }
            leaf tiebreaker {
                type mtt:path-tiebreaker-type;
            }
        }
    }

    grouping signaling-properties {
        description "LSP signaling properties.";
        leaf source {
            description
                "LSP source address.";
            type inet:ip-address;
        }
        container fast-reroute {
            presence "Requests FRR local protection on LSRs";
            leaf bandwidth-protect {
                description
                    "Requests FRR bandwidth protection on LSRs";
                type boolean;
            }
            leaf node-protect {
                description
                    "Requests FRR node protection on LSRs";
                type boolean;
            }
        }
        leaf record-route {
            description
                "Requests path RRO recording in RSVP PATH message.";
            type boolean;
        }
        leaf signaled-name {
            description
                "Sets the session name to use in the session attribute
                 object.";
            type string;
        }
        container priority {
            description
                "Sets the setup/hold priority to use in the session
                 attribute object";
            leaf setup {
                type uint8;
            }
            leaf hold {
                type uint8;
            }
        }
        leaf soft-preemption {
            description
                "Requests soft-preemption in session
                 attributes object at
                 at traversed LSR(s).";
            type boolean;
        }
        container signaled-bandwidth {
            description
                "Sets the requested bandwidth";
            leaf type {
                type mtt:bandwidth-type;
            }
            leaf value {
                type uint32;
            }
        }
    }

    grouping logging-events {
        container logging-events {
            choice type {
                container all {
                    presence "Enable all MPLS-TE tunnel event logging.";
                }
                container bfd-status {
                    presence "Enable MPLS-TE tunnel BFD specific event
                              logging.";
                }
                container link-status {
                    presence "Enable link status event logging.";
                }
                container lsp-status {
                    presence "Enable LSP status event logging.";
                    leaf all {
                        type boolean;
                    }
                    leaf-list events {
                        type mtt:lsp-status-event-type;
                    }
                }
                container cspf-failure {
                    presence "Enable MPLS-TE tunnel CSPF failure event
                              logging.";
                }
            }
        }
    }

    grouping affinity-properties {
        choice style {
            container affinity-hex {
                leaf value {
                    type uint32;
                }
                leaf mask {
                    type uint32;
                }
            }
            container affinity-name {
                list constraints {
                    key "action";
                    leaf action {
                        type mtt:affinity-action-type;
                    }
                    container constraint {
                        list affinity-list {
                            key "name";
                            leaf name {
                                type string;
                            }
                        }
                    }
                }
            }
        }
    }

    grouping routing-proprties {
        choice routing-proprties {
            description
                "Announces the tunnel to IGP as either autoroute or
                 forwarding adjacency.";
            container autoroute {
                presence "Enables autoroute announce.";
                description
                    "Announce the MPLS-TE tunnel as autoroute to IGP for
                     use as IGP shortcut";
                leaf-list routing-afs {
                     type inet:ip-version;
                }
                choice metric-type {
                    leaf metric {
                        type uint32;
                    }
                    leaf relative-metric {
                        type uint32;
                    }
                    leaf absolute-metric {
                        type uint32;
                    }
                }
            }
            container forwarding-adjacency {
                presence "Enables forwarding adjacency on the tunnel.";
                description
                    "Announce the MPLS-TE tunnel as
                     forwarding adjacency.";
                leaf holdtime {
                    description
                        "Holdtime in seconds after tunnel becomes UP.";
                    type uint32;
                }
                leaf-list routing-afs {
                     type inet:ip-version;
                }
            }
        }
    }

    grouping auto-bandwidth {
        container auto-bandwidth {
            presence "Enabled MPLS-TE tunnel auto-bandwidth feature";
            description
                "MPLS-TE tunnel auto-bandwidth configuration
                parameters.";
            leaf overflow-threshold {
                description
                    "Auto-bandwidth change percent to trigger overflow";
                type uint32;
            }
            leaf overflow-limit {
                description
                    "Auto-bandwidth consecutive collections to trigger
                      overflow";
                type uint8;
            }
            leaf underflow-threshold {
                description
                    "Auto-bandwidth change percent to
                     trigger underflow";
                type uint32;
            }
            leaf underflow-limit {
                description
                    "Auto-bandwidth consecutive collections to trigger
                      underflow";
                type uint8;
            }
            leaf collect-only {
                description
                    "Auto-bandwidth collection only mode";
                type boolean;
            }
            leaf application-frequency {
                description
                    "Auto-bandwidth application interval in seconds";
                type uint32;
            }
            container bandwidth-limit {
                leaf min-limit {
                    type uint32;
                }
                leaf max-limit {
                    type uint32;
                }
            }
        }
    }

    grouping backup-bandwidth {
        container backup-bandwidth {
            when "/tunnels-cfg/tunnel/backup = true";
            description
                "The backup bandwidth that can be protected by this
                 backup tunnel.";
            leaf value {
                type uint32;
            }
            leaf type {
                type mtt:backup-bandwidth-type;
            }
        }
    }

    grouping forwarding-properties {
        description "Policy for using tunnel in forwarding.";
        choice forwarding-property {
            container forwarding-class {
                leaf class {
                    type uint8;
                    default 0;
                }
            }
            container forwarding-group {
                leaf-list classes {
                    type uint8;
                }
            }
        }
    }

    grouping flooding-thresholds {
        description "Flooding threshold values.";
        container flooding-thresholds {
            choice type {
                case single-step {
                    leaf up-step { type uint8; }
                    leaf down-step { type uint8; }
                }
                case multiple-steps {
                    list up-steps {
                        key "value";
                        description "Percentage bandwidth
                            exceeded that causes flooding";
                        leaf value { type uint8; }
                    }
                    list down-steps {
                        key "value";
                        description "Percentage bandwidth
                               crossed that causes flooding";
                            leaf value { type uint8; }
                    }
                }
            }
        }
    }
    /* MPLS-TE Tunnel Template Configuration Data */
    grouping tunnel-template {
        leaf name {
            description "MPLS-TE tunnel-template name.";
            type string;
        }
        uses signaling-properties;
        uses affinity-properties;
        uses logging-events;
        uses auto-bandwidth;
        uses routing-proprties;
        uses forwarding-properties;
    }

    /* MPLS-TE Tunnel Configuration Data */
    container tunnels-cfg {
        description
            "Configuration, operational, notification,
              and RPC data model
             for MPLS-TE tunnels.";
        presence "Enables MPLS-TE Global mode.";
        list tunnel {
            key "name type";
            unique "identifier";
            description "MPLS-TE tunnel.";
            leaf name {
                type string;
                description "MPLS-TE tunnel name.";
            }
            leaf type {
                description "MPLS-TE tunnel type.";
                type mtt:tunnel-type;
            }
            leaf identifier {
                description
                    "MPLS-TE tunnel Identifier.";
                type uint16;
            }
            leaf description {
                description
                    "MPLS-TE tunnel description.";
                type string;
            }
            leaf admin-mode {
                description "MPLS-TE tunnel administrative state";
                type enumeration {
                    enum up;
                    enum down;
                }
                default up;
            }
            leaf path-protection {
                description "Enable MPLS-TE tunnel
                     end-to-end path-protection.";
                type boolean;
            }
            leaf backup {
                description "Tunnel is being used
                       as fast reroute backup.";
                type boolean;
                default false;
            }
            leaf load-share {
                description "ECMP tunnel forwarding load-share factor.";
                type uint32;
            }
            uses auto-bandwidth;
            uses forwarding-properties;
            uses routing-proprties;
            uses backup-bandwidth;
            container bidirectional {
                description
                    "MPLS-TE associated
                             bidirectional tunnel attributes.";
                container association {
                    leaf id {
                        description
                            "The MPLS-TE tunnel
                              association identifier.";
                        type uint32;
                    }
                    leaf source {
                        description
                            "The MPLS-TE tunnel association source.";
                        type inet:ip-address;
                    }
                    leaf global-source {
                        description
                            "The MPLS-TE tunnel association global
                             source.";
                        type inet:ip-address;
                    }
                    leaf type {
                        description "The MPLS-TE
                               tunnel association type.";
                        type mtt:bidir-association-type;
                    }
                }
            }

            /* List of destinations and path(s) */
            list destination {
                key "address";
                description
                    "The MPLS-TE tunnel destination address.";
                leaf address {
                    type inet:ip-address;
                }
                list paths {
                    key "index";
                    leaf index {
                        type uint8;
                    }
                    uses path-properties;
                }
            }

            uses lsp-properties;
        }
    }

    /* MPLS-TE Global Configuration Data */
    container global-cfg {

        description
            "Configuration data model for Global System-wide MPLS
             Traffic Engineering.";
        presence "Enables MPLS-TE Global mode.";

        grouping load-share-properties {
            container load-share-properties {
                description "ECMP tunnel forwarding
                            load-share properties.";
                container unequal {
                    presence "ECMP tunnels to
                             load-share forwarding of
                             traffic unequally.";

                    leaf bandwidth-based {
                         type boolean;
                         description
                             "ECMP tunnels to load-share
                             forwarding of traffic
                             based on tunnel bandwidth.";
                    }
                }
            }
        }

        grouping global-timers {
            container global-timers {
                description
                    "Global system-wide TE timer values.";
                leaf lsp-bw-hold-delay {
                     type  uint32 ;
                     description
                         "Bandwidth hold interval for
                         LSP admission in seconds.";
                }
                leaf lsp-preemption-delay {
                     type  uint32 ;
                     description
                         "Delay in seconds to preempt
                          LSPs after preemption
                         event triggered.";
                }
                leaf topology-holddown-sigerr {
                     type  uint32 ;
                     description
                         "Link holddown in topology database for CSPF.
                         Delay in seconds after signaling error.";
                }
            }
        }

        grouping global-reoptimization {
            description
                "Global periodic LSP reoptimization parameters.";
            container global-reoptimization {
                presence "Enable TE tunnel reoptimization globally
                         on the node.";

                leaf interval {
                     type  uint32;
                     description
                         "Periodic reoptimization
                          interval in seconds.";
                }
                leaf installation-delay {
                     type  uint32;
                     description
                         "Delay in seconds before
```

```
                          installing reoptimzing LSP
                         in forwarding to carry traffic.";
                }
                leaf cleanup-delay {
                     type  uint32;
                     description
                         "Delay in seconds before
                          removing reoptimized LSP
                         in forwarding.";
                }
            }
        }

        grouping topology-flooding {
            description
                "Global periodic TE topology flooding parameters.";
            container topology-flooding {
                presence "Enable TE topology
                flooding globally on the node.";

                leaf flooding-interval {
                     type  uint32 ;
                     description
                         "Periodic topology flooding
                         interval in seconds.";
                }

                uses flooding-thresholds;
            }
        }

        grouping affinity-map {
            description
                "Mapping of affinity name and value.";
            leaf name {
                description
                    "Name of the affinity.";
                type string;
            }
            choice type  {
                leaf value {
                    type uint32;
                }
                /* TBA: 256 bit value */
                leaf bit-index {
                    type uint32;
                }
            }
        }

        grouping affinity-maps {
            description
                "Mapping of Affinity name and value.";

            container affinity-maps {
                list affinity-maps {
                    description "MPLS-TE affinity-maps.";
                    key "name";
                    uses affinity-map;
                }
            }
        }

        grouping srlg-map {
            description
                "Mapping of SRLG name, value and admin-weight.";
            leaf name {
                type string;
            }
            leaf value {
                type uint32;
            }
            leaf admin-weight {
                type uint32;
            }
        }

        grouping srlg-maps {
            description
                "Mapping of SRLG name, value and admin-weight.";
            container srlg-maps {
                list srlg-maps {
                    description "MPLS-TE srlg-map.";
                    key "name";
                    uses srlg-map;
                }
            }
        }

        grouping softpreemption-properties {
            description
                "Softpreemption properties.";
            container softpreemption-te {
                presence "Enable soft-preemption on the node.";

                leaf softpremption-timeout {
                     type  uint32 ;
                     description
                         "Delay in seconds to teardown soft-preempted
                         LSPs after soft-preemption event.";
                }
            }
        }

        grouping diff-serve-te-properties {
            description
                "Diff-Serve TE properties.";
            container ds-te {
                presence "Enable Diff-Serve TE on the node.";

                leaf bc-model {
                     description
                         "Diff-Serve TE bandwidth
                          constraint model type.";
                     type mtt:bc-model-type;
                }
                list te-class {
                    description
                        "Diff-Serve TE TE-class mapping.";
                    key "class-index";
                    leaf class-index {
                         description
                            "TE-class index.";
                         type uint8;
                    }
                    leaf priority {
                         description
                             "LSP setup or hold priority.";
                         type uint8;
                    }
                    leaf bc-value {
                         description
                             "Bandwidth Constraint pool value.";
                         type uint8;
                    }
                }
            }
        }

        grouping auto-bandwidth-properties {
            description
                "Auto-bandwidth adjustment properties.";
            container auto-bw {
                presence "Enable auto-bandwidth on the node.";
                leaf stats-collection-interval {
                    description
                        "Auto-bandwidth statistics collection
                        interval in seconds.";
                    type uint32;
                }
            }
        }

        grouping auto-mesh-properties {
            description
                "Auto-tunnel mesh-group properties.";
            leaf mesh-group {
                 description
                    "Value of the mesh-group.";
                 type uint32;
            }
            leaf template-name {
                 description
                    "Name of the template to use
                     for tunnel properties.";
                 type string;
            }
            leaf one-hop-only {
                 description
                    "Limit tunnel to one-hop away nodes only.";
                 type boolean;
            }
        }

        grouping auto-backup-properties {
            description
                "Auto-tunnel backup properties.";
            leaf template-name {
                 description
                    "Name of the template to use for
                     tunnel properties.";
                 type string;
            }
            leaf nhop-only {
                 description
                    "Limit tunnel to link protection only.";
                 type boolean;
            }
        }

        grouping tail-signaling {
            description
```

```
                "Signaling properties for LSP tail-end.";
            container tail-signaling {
                leaf advertise-explicit-null {
                     type boolean;
                }
            }
        }

        uses path-computation-properties;
        uses bfd-properties;
        uses logging-events;

        /* template-type is a leaf. mtt:tunnel-type is also leaf.
         */
        list tunnel-templates {
            description "MPLS-TE templates.";
            key "name";
            uses tunnel-template;
        }

        list auto-tunnel-mesh {
            description "MPLS-TE auto-tunnel mesh-groups.";
            key "mesh-group";
            uses auto-mesh-properties;
        }

        uses auto-backup-properties;
        uses load-share-properties;
        uses global-timers;
        uses topology-flooding;
        uses global-reoptimization;
        uses softpreemption-properties;
        uses affinity-maps;
        uses srlg-maps;
        uses diff-serve-te-properties;
        uses auto-bandwidth-properties;
        uses tail-signaling;
    }

    /* MPLS-TE Link Configuration Data */
    container links-cfg {
        description
            "Configuration data model for MPLS-TE links.";

        typedef  auto-backup-path-computation-type {
            type enumeration {
                enum srlg-none;
                enum srlg-strict;
                enum srlg-preferred;
                enum srlg-weighted;
            }
        }

        grouping bandwidth-constraints {
            container bandwidth-constraints {
                leaf maximum-reservable {type uint32;}
                leaf bc0 {type uint32;}
                leaf bc1 {type uint32;}
                leaf bc2 {type uint32;}
                leaf bc3 {type uint32;}
                leaf bc4 {type uint32;}
                leaf bc5 {type uint32;}
                leaf bc6 {type uint32;}
                leaf bc7 {type uint32;}
            }
        }

        list link {
            key "name";
            description "MPLS-TE links.";
            leaf name {
                type string;
                description "MPLS-TE link name.";
            }

            leaf admin-weight {
                description "MPLS-TE admin-weight.";
                type uint32;
            }

            container reservable-bandwidths {
                description "Reservabale bandwidth values.";
                leaf bc-model {
                     description
                         "Diff-Serve TE bandwidth
                         constraint model type.";
                     type mtt:bc-model-type;
                }
                leaf value-type {
                     /* absolute-value */
                     /* percentage-value */
                     type boolean;
                }
                uses bandwidth-constraints;
            }
            uses flooding-thresholds;

            container link-affinities {
                choice type {
                    case bitmap {
                        leaf bitmap {
                             type uint32;
                        }
                    }
                    case bitmap-extended {
                        /* TBA: 256 bit */
                        leaf bitmap-extended {
                             type uint32;
                        }
                    }
                    case names {
                        list names {
                            key "name";
                            description "List of affinity names.";
                            leaf name {
                                type string;
                            }
                        }
                    }
                }
            }

            list shared-risk-link-group {
                key "name";
                description "List of SRLGs that this link is part of";
                leaf name {
                     type string;
                }
            }

            container fast-reroute-backup {
                choice type {
                    case configured-backups {
                        list configured-backups {
                            key "name";
                            description "List of backup
                            tunnels to protect this link";
                            leaf name {type string;}
                        }
                    }
                    case auto-backup {
                        uses tunnel-template;
                        leaf link-protection-only {
                             type boolean;
                        }
                        leaf bandwidth-protection {
                             type boolean;
                        }
                        leaf backup-path-computation {
                            type auto-backup-path-computation-type;
                        }
                    }
                }
            }
        }
    }


    /* MPLS-TE Tunnel Operational Data */
    container tunnels-oper {
        config "false";
        /* mandatory "true"; */
        description "MPLS-TE tunnel operational state data.";
        list tunnel {
            description "MPLS-TE tunnel.";
            key "name";
            leaf name {
                type string;
                description "MPLS-TE tunnel name.";
            }
        }
    }

    /* MPLS-TE Global Operational Data */
    container global-oper {
        config "false";
    }

    /* MPLS-TE Global Operational Data */
    container links-oper {
        description
            "Operational data model for MPLS-TE links.";
        config "false";
    }

    /* MPLS-TE Tunnel RPCs/execution Data */
    rpc tunnels-rpc {
        description
            "foo bar";
        input {
            leaf input-command {
                type string;
                default "";
                description
                    "The string with which the tape shall
                    be initialized. The
                    leftmost symbol will be at tape
                    coordinate 0.";
            }
        }
        output {
            leaf out-message {
                type string;
                default "";
                description "foo bar";
            }
        }
    }

    /* MPLS-TE Global RPCs/execution Data */
    rpc global-rpc {
        description
            "Execution data for MPLS-TE global.";
    }

    /* MPLS-TE links RPCs/execution Data */
    rpc links-rpc {
        description
            "Execution data for MPLS-TE links.";
    }

    /* MPLS-TE Tunnel Notification Data */
    notification tunnels-notif {
        description
            "The Turing Machine has halted. This means that there is no
            transition rule for the current state and tape symbol.";
        leaf state {
            type mtt:tunnel-type;
            mandatory "true";
        }
    }

    /* MPLS-TE Global Notification Data */
    notification global-notif {
    }

    /* MPLS-TE Links Notification Data */
    notification links-notif {
    }
```

```
}
```

12. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

```
URI: urn:ietf:params:xml:ns:yang:ietf-mpls-te
XML: N/A, the requested URI is an XML namespace.
```

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name:       ietf-mpls-te
namespace:  urn:ietf:params:xml:ns:yang:ietf-mpls-te
prefix:     mpls-te
reference:  RFC XXXX
```

13. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations. Following are the subtrees and data nodes and their sensitivity/vulnerability:

/global-cfg: This module specifies the global MPLS-TE configurations on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/tunnels-cfg/tunnel: This list specifies the configured MPLS-TE tunnes on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/lsps-cfg/lsp: This list specifies the configured MPLS-TE LSPs on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/links-cfg/link: This list specifies the configured MPLS-TE links on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

14. Acknowledgement

TBA.

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

15.2. Informative References

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A.

Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[I-D.ietf-netmod-routing-cfg] Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-16 (work in progress), May 2014.

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-01 (work in progress), June 2014.

16. Authors’ Addresses

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Tarek Saad
Cisco Systems, Inc.

Email: tsaad@cisco.com

Robert Sawaya
Cisco Systems, Inc.

Email: rsawaya@cisco.com

Internet Engineering Task Force
Internet-Draft
Updates: 5884 (if approved)
Intended status: Standards Track
Expires: April 6, 2014

V. Govindan
Cisco Systems
K. Rajaraman
G. Mirsky
Ericsson
N. Akiya
Cisco Systems
S. Aldrin
Huawei Technologies

October 3, 2014

Clarifications to RFC 5884
draft-grmas-bfd-rfc5884-clarifications-00

Abstract

This document clarifies the procedures for establishing, maintaining and removing multiple, concurrent BFD sessions for a given <MPLS LSP, FEC> described in RFC5884.

Status of This Memo

Copyright Notice

Table of Contents

1. Background

[RFC5884] defines the procedures to bootstrap and maintain BFD sessions for a <MPLS FEC, LSP> using LSP ping. While Section 4 of [RFC5884] specifies that multiple BFD sessions can be established for a <MPLS FEC, LSP> tuple, the procedures to bootstrap and maintain multiple BFD sessions concurrently over a <MPLS FEC, LSP> are not clearly specified. Additionally, the procedures of removing BFD sessions bootstrapped on the egress LSR are unclear. This document provides those clarifications without deviating from the principles outlined in [RFC5884].

The ability for an ingress LSR to establish multiple BFD sessions for a <MPLS FEC, LSP> tuple is useful in scenarios such as Segment Routing based LSPs or LSPs having Equal-Cost Multipath (ECMP). The process used by the ingress LSR to determine the number of BFD session(s) to be bootstrapped for a <MPLS FEC, LSP> tuple and the mechanism of constructing those session(s) are outside the scope of this document.

1.1. Requirements Language

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

2. Theory of Operation

2.1. Procedures for establishment of multiple BFD sessions

Section 6 of [RFC5884] specifies the procedure for bootstrapping BFD sessions using LSP ping. It further states that a BFD session SHOULD be established for each alternate path that is discovered. This requirement has been the source of some ambiguity as the procedures of establishing concurrent, multiple sessions have not been explicitly specified. This ambiguity can also be attributed in part to the text in Section 7 of [RFC5884] forbidding either end to change local discriminator values in BFD control packets after the session reaches the UP state. The following procedures are described to clarify the ambiguity based on the interpretation of the authors's reading of the referenced sections:

At the ingress LSR:

> MPLS LSP ping can be used to bootstrap multiple BFD sessions for a given <MPLS FEC, LSP>. Each LSP ping MUST carry a different discriminator value in the BFD discriminator TLV [RFC4379].

The egress LSR needs to perform the following:

> If the validation of the FEC in the MPLS Echo request message succeeds, check the discriminator specified in the BFD discriminator TLV of the MPLS Echo request. If there is no local session that corresponds to the discriminator (remote) received in the MPLS Echo request, a new session is bootstrapped and a local discriminator is allocated. Since the BFD local discriminator of either ends cannot change as long as the session is in the UP state, a new discriminator received in the LSP ping unambiguously conveys the intent of the LSR ingress to bootstrap a new BFD session for the FEC specified in the LSP ping.
>
> Ensure the uniqueness of the <MPLS FEC, LSP, Remote Discriminiator> tuple.
>
> The remaining procedures of session establishment are as specified in [RFC5884].

2.2. Procedures for maintenance of multiple BFD sessions

Both the ingress LSR and egress LSR use the YourDiscriminator of the received BFD packet to demultiplex BFD sessions.

2.3. Procedures for removing BFD sessions at the egress LSR

[RFC5884] does not specify an explicit procedure for deleting BFD sessions. The procedure for removing a BFD session established by an out-of-band discriminator exchange using the MPLS LSP ping can improve resource management (like memory etc.) especially in scenarios involving thousands or more of such sessions. A few options are possible here:

> The BFD session MAY be removed in the egress LSR if the BFD session transitions from UP to DOWN. This can be done after the expiry of a configurable timer started after the BFD session state transitions from UP to DOWN at the egress LSR.
>
> The BFD session on the egress LSR MAY be gracefully removed by the ingress LSR by using the BFD diagnostic code AdminDown(7) specified in [RFC5880]. When the ingress LSR wants to gracefully remove a session, it MAY transmit BFD packets containing the diagnostic code AdminDown(7) detectMultiplier number of times. Upon receiving such a packet, the egress LSR MAY remove the BFD session gracefully, without triggering a change of state.

Ed Note: The procedures to be followed at the egress LSR when the BFD session never transitions to UP from DOWN state are yet to be clarified

Regardless of the option chosen to proceed, all BFD sessions established with the FEC MUST be removed automatically if the FEC is removed.

2.4. Changing discriminators for a BFD session

The discriminators of a BFD session established over an MPLS LSP cannot be changed when it is in UP state. The BFD session could be removed after a graceful transition to AdminDown state using the BFD diagnostic code AdminDown. A new session could be established with a different discriminator. The initiation of the transition from the Up to Down state can be done either by the ingress LSR or the egress LSR.

3. Backwards Compatibility

The procedures clarified by this document are fully backward compatible with an existing implementation of [RFC5884]. While the capability to bootstrap and maintain multiple BFD sessions may not be present in current implementations, the procedures outlined by this document can be implemented as a software upgrade without affecting existing sessions. In particular, the egress LSR needs to support multiple BFD sessions per <MPLS FEC, LSP> before the ingress LSR is upgraded.

4. Encapsulation

The encapsulation of BFD packets are the same as specified by [RFC5884].

5. Security Considerations

This document clarifies the mechanism to bootstrap multiple BFD sessions per <MPLS FEC, LSP>. BFD sessions, naturally, use system and network resources. More BFD sessions means more resources will be used. It is highly important to ensure only minimum number of BFD sessions are provisioned per FEC, and bootstrapped BFD sessions are properly deleted when no longer required. Additionally security measures described in [RFC4379] and [RFC5884] are to be followed.

6. IANA Considerations

This document does not make any requests to IANA.

7. Acknowledgements

The authors would like to thank Mudigonda Mallik, Rajaguru Veluchamy and Carlos Pignataro of Cisco Systems for their review comments.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

Authors’ Addresses

Vengada Prasad Govindan
Cisco Systems

Email: venggovi@cisco.com

Kalyani Rajaraman
Ericsson

Email: kalyani.rajaraman@ericsson.com

Gregory Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Nobo Akiya
Cisco Systems

Email: nobo@cisco.com

Sam Aldrin
Huawei Technologies

Email: aldrin.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2015

X. Xu
Huawei Technologies
N. Sheth
Juniper Networks
L. Yong
Huawei USA
C. Pignataro
Cisco Systems
Y. Fan
China Telecom
R. Callon
Juniper Networks
D. Black
EMC Corporation

October 24, 2014

Encapsulating MPLS in UDP
draft-ietf-mpls-in-udp-07

Abstract

This document specifies an IP-based encapsulation for MPLS, called MPLS-in-UDP (User Datagram Protocol). The MPLS-in-UDP encapsulation technology MUST only be deployed within a service provider network or networks of an adjacent set of co-operating service providers where congestion control is not a concern.

Status of This Memo

This Internet-Draft will expire on April 27, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an IP-based encapsulation for MPLS, i.e. MPLS-in-UDP (User Datagram Protocol), which is applicable in some circumstances where IP-based encapsulation for MPLS is required and further fine-grained load balancing of MPLS packets over IP networks over Equal Cost Multi-Path (ECMP) and/or Link Aggregation Groups (LAG) is required as well. There are already IP-based encapsulations for MPLS that allow for fine-grained load balancing by using some special field in the encapsulation header as an entropy field. However, MPLS-in-UDP can be advantageous since some networks have used the UDP port number fields as a basis for load-balancing solutions for some time. This is similar to why LISP [RFC6830] uses UDP encapsulation.

Like other IP-based encapsulation methods for MPLS, this encapsulation allows for two Label Switching Routers (LSR) to be adjacent on a Label Switched Path (LSP), while separated by an IP network. In order to support this encapsulation, each LSR needs to know the capability to decapsulate MPLS-in-UDP by the remote LSRs. This specification defines only the data plane encapsulation and does not concern itself with how the knowledge to use this encapsulation is conveyed. Specifically, this capability can be either manually configured on each LSR or be dynamically advertised in manners that are outside the scope of this document.

Similarly, the MPLS-in-UDP encapsulation format defined in this document by itself cannot ensure the integrity and privacy of data packets being transported through the MPLS-in-UDP tunnels and cannot enable the tunnel decapsulators to authenticate the tunnel encapsulator. Therefore, in the case where any of the above security issues is concerned, the MPLS-in-UDP SHOULD be secured with IPsec [RFC4301] or DTLS [RFC6347]. For more details, please see Section 6 of Security Considerations.

1.1. Existing Encapsulations

Currently, there are several IP-based encapsulations for MPLS such as MPLS-in-IP, MPLS-in-GRE (Generic Routing Encapsulation) [RFC4023], and MPLS-in-L2TPv3 (Layer Two Tunneling Protocol - Version 3) [RFC4817]. In all these methods, the IP addresses can be varied to achieve load-balancing.

All these IP-based encapsulations for MPLS are specified for both IPv4 and IPv6. In the case of IPv6-based encapsulations, the IPv6 Flow Label can be used for ECMP and LAGs [RFC6438]. However, there is no such entropy field in the IPv4 header.

For MPLS-in-GRE as well as MPLS-in-L2TPv3, protocol fields (the GRE Key and the L2TPv3 Session ID respectively) can be used as the load-balancing key as described in [RFC5640]. For this, intermediate routers need to understand these fields in the context of being used as load-balancing keys.

1.2. Motivations for MPLS-in-UDP Encapsulation

Most existing routers in IP networks are already capable of distributing IP traffic "microflows" [RFC2474] over ECMPs and/or LAG based on the hash of the five-tuple of User Datagram Protocol (UDP) [RFC0768] and Transmission Control Protocol (TCP) packets (i.e., source IP address, destination IP address, source port, destination port, and protocol). By encapsulating the MPLS packets into an UDP tunnel and using the source port of the UDP header as an entropy field, the existing load-balancing capability as mentioned above can be leveraged to provide fine-grained load-balancing of MPLS traffic over IP networks.

1.3. Application Statements

The MPLS-in-UDP encapsulation technology MUST only be deployed within a Service Provider (SP) network or networks of an adjacent set of co-operating SPs where congestion control is not a concern, rather than over the Internet where congestion control is required. Furthermore, packet filters SHOULD be added to prevent MPLS-in-UDP packets from escaping from the service provider networks due to misconfiguation or packet errors.

2. Terminology

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

3. Encapsulation in UDP

MPLS-in-UDP encapsulation format is shown as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Source Port = Entropy      |       Dest Port = MPLS        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           UDP Length          |        UDP Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                        MPLS Label Stack                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                |
                                |
~                         Message Body                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Port of UDP

This field contains a 16-bit entropy value that is generated by the encapsulator to uniquely identify a flow. What constitutes a flow is locally determined by the encapsulator and therefore is outside the scope of this document. What algorithm is actually used by the encapsulator to generate an entropy value is outside the scope of this document.

In case the tunnel does not need entropy, this field of all packets belonging to a given flow SHOULD be set to a randomly selected constant value so as to avoid packet reordering.

To ensure that the source port number is always in the range 49152 to 65535 (Note that those ports less than 49152 are reserved by IANA to identify specific applications/protocols) which may be required in some cases, instead of calculating a 16-bit hash, the encapsulator SHOULD calculate a 14-bit hash and use those 14 bits as the least significant bits of the source port field while the most significant two bits SHOULD be set to binary 11. That still conveys 14 bits of entropy information which would be enough as well in practice.

Destination Port of UDP

This field is set to a value (TBD1) allocated by IANA to indicate that the UDP tunnel payload is an MPLS packet.

UDP Length

The usage of this field is in accordance with the current UDP specification [RFC0768].

UDP Checksum

For IPv4 UDP encapsulation, this field is RECOMMENDED to be set to zero because the IPv4 header includes a checksum and use of the UDP checksum is optional with IPv4, unless checksum protection of VPN labels is important (See Section 6). For IPv6 UDP encapsulation, the IPv6 header does not include a checksum, so this field MUST contain a UDP checksum that MUST be used as specified in [RFC0768] and [RFC2460] unless one of the exceptions that allows use of UDP zero-checksum mode (as specified in [RFC6935]) applies. See Section 3.1 for specification of these exceptions and additional requirements that apply when UDP zero-checksum mode is used for MPLS-in-UDP traffic over IPv6.

MPLS Label Stack

This field contains an MPLS Label Stack as defined in [RFC3032].

Message Body

This field contains one MPLS message body.

3.1. UDP Checksum Usage with IPv6

When UDP is used over IPv6, the UDP checksum is relied upon to protect the IPv6 header from corruption, and MUST be used unless the requirements in [RFC6935] and [RFC6936] for use of UDP zero-checksum mode with a tunnel protocol are satisfied. MPLS-in-UDP is a tunnel protocol, and there is significant successful deployment of MPLS-in-IPv6 without UDP (i.e., without a checksum that covers the IPv6 header or the MPLS label stack), as described in Section 3.1 of [RFC6936]:

> "There is extensive experience with deployments using tunnel protocols in well-managed networks (e.g., corporate networks and service provider core networks). This has shown the robustness of methods such as Pseudowire Emulation Edge-to-Edge (PWE3) and MPLS that do not employ a transport protocol checksum and that have not specified mechanisms to protect from corruption of the unprotected headers(such as the VPN Identifier in MPLS)".

This draft focuses on service provider core networks. The requirements in Section 5 for use of MPLS-in-UDP to carry traffic that is not necessarily congestion controlled involve significant service provider traffic management and engineering - this is a hallmark of the well-managed networks that the above [RFC6936] text refers to. Therefore, the UDP checksum MUST be implemented and MUST be used in accordance with [RFC0768] and [RFC2460] for MPLS-in-UDP traffic over IPv6 unless one of the following exceptions applies and the additional requirements stated below are complied with. There are two exceptions that allow use of UDP zero-checksum mode for IPv6 with MPLS-in-UDP, subject to the additional requirements stated below in this section. The two exceptions are:

a. Use of MPLS-in-UDP within a single service provider that utilizes careful provisioning (e.g., rate limiting at the entries of the network while over-provisioning network capacity) to ensure against congestion and that actively monitors MPLS traffic for errors; or

b. Use of MPLS-in-UDP within a limited number of service providers who closely cooperate in order to jointly provide this same careful provisioning and monitoring.

Even when one of the above exceptions applies, use of UDP checksums may be appropriate when VPN services are provided over MPLS-in-UDP, see Section 6. As such, for IPv6, the UDP checksum for MPLS-in-UDP MUST be used as specified in [RFC0768] and [RFC2460] over the general

Internet, and over non-cooperating SPs, even if each non-cooperating SP independently satisfies the first exception for UDP zero-checksum mode usage with MPLS-in-UDP over IPv6 within the SP’s own network. Measures SHOULD be taken to prevent UDP zero checksum mode MPLS-in-UDP over IPv6 traffic from "escaping" to the general Internet; see Section 5 for examples of such measures.

The following additional requirements apply to implementation and use of UDP zero-checksum mode for MPLS-in-UDP over IPv6:

a. Use of the UDP checksum with IPv6 MUST be the default configuration of all MPLS-in-UDP implementations.

b. An MPLS-in-UDP implementation MUST comply with all requirements specified in Section 4 of [RFC6936] and with requirement 1 specified in Section 5 of [RFC6936].

c. An MPLS-in-UDP receiver MUST check that the source and destination IPv6 addresses are valid for the MPLS-in-UDP tunnel and discard any packet for which this check fails.

d. An MPLS-in-UDP sender SHOULD use different IPv6 addresses for each MPLS-in-UDP tunnel that uses UDP zero-checksum mode in order to strengthen the receiver’s check of the IPv6 source address. When this is not possible, it is RECOMMENDED to use each source IPv6 address for as few UDP zero-checksum mode MPLS-in-UDP tunnels as is feasible.

e. An MPLS-in-UDP receiver MUST check that the top label of the MPLS label stack is valid for the tunnel. This check will often be part of the MPLS LSR forwarding logic, but MUST be scoped to the specific tunnel.

f. An MPLS-in-UDP receiver node SHOULD only enable the use of UDP zero-checksum mode on a single UDP port and SHOULD NOT support any other use UDP zero-checksum mode on any other UDP port.

g. Any middlebox support for MPLS-in-UDP with UDP zero-checksum mode for IPv6 MUST comply with requirements 1 and 8-10 in Section 5 of [RFC6936].

The above requirements do not change the requirements specified in [RFC2460] as modified by [RFC6935] and the requirements specified in [RFC6936].

The requirements to check the source IPv6 address and top label of the MPLS stack (in addition to the destination IPv6 address), plus the strong recommendation against reuse of source IPv6 addresses among MPLS-in-UDP tunnels collectively provide some offset for the absence of UDP checksum coverage of the IPv6 header. The expected result for IPv6 UDP zero-checksum mode for MPLS-in-UDP is that corruption of the destination IPv6 address will usually cause packet discard, as offsetting corruptions to the source IPv6 and/or MPLS top label are unlikely. Additional assurance is provided by the restrictions in the above exceptions that limit usage of IPv6 UDP zero-checksum mode to specific types of well-managed networks for which MPLS packet corruption has not been a problem in practice.

Hence MPLS-in-UDP is suitable for transmission over lower layers in the well-managed networks that are allowed by the two exceptions stated above and is not expected to increase the rate of corruption of the inner IP packet on such networks by comparison to MPLS traffic that is not encapsulated in UDP. For these reasons, MPLS-in-UDP does not provide an additional integrity check when UDP zero-checksum mode is used with IPv6, and this design is in accordance with requirements 2, 3 and 5 specified in Section 5 of [RFC6936].

MPLS does not accumulate incorrect state as a consequence of label stack corruption. A corrupt MPLS label results in either packet discard or forwarding (and forgetting) of the packet without accumulation of MPLS protocol state. Active monitoring of MPLS-in-UDP traffic for errors is REQUIRED as occurrence of errors will result in some accumulation of error information outside the MPLS protocol for operational and management purposes. This design is in accordance with requirement 4 specified in Section 5 of [RFC6936].

The remaining requirements specified in Section 5 of [RFC6936] are inapplicable to MPLS-in-UDP. Requirements 6 and 7 do not apply because MPLS does not have an MPLS-generic control feedback mechanism. Requirements 8-10 are middlebox requirements that do not apply to MPLS-in-UDP tunnel endpoints, but see Section 3.2 for further middlebox discussion.

In summary, UDP zero-checksum mode for IPv6 is allowed to be used with MPLS-in-UDP when one of the two exceptions specified above applies, provided that the five additional requirements (six for middlebox implementations) stated above are complied with. Otherwise the UDP checksum MUST be used for IPv6 as specified in [RFC0768] and [RFC2460].

This entire section and its requirements apply only to use of UDP zero-checksum mode for IPv6; they can be avoided by using the UDP checksum as specified in [RFC0768] and [RFC2460].

3.2. Middlebox Considerations for IPv6 UDP Zero Checksums

IPv6 datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum based on [RFC2460] or that updates the UDP checksum field, such as NATs or firewalls. Changing this behavior would require such middleboxes to be updated to correctly handle datagrams with zero UDP checksums. The MPLS-in-UDP encapsulation does not provide a mechanism to safely fall back to using a checksum when a path change occurs redirecting a tunnel over a path that includes a middlebox that discards IPv6 datagrams with a zero UDP checksum. In this case the MPLS-in-UDP tunnel will be black-holed by that middlebox. Recommended changes to allow firewalls, NATs and other middleboxes to support use of an IPv6 zero UDP checksum are described in Section 5 of [RFC6936].

4. Processing Procedures

This MPLS-in-UDP encapsulation causes MPLS packets to be forwarded through "UDP tunnels". When performing MPLS-in-UDP encapsulation by the encapsulator, the entropy value would be generated by the encapsulator and then be filled in the Source Port field of the UDP header. The Destination Port field is set to a value (TBD1) allocated by IANA to indicate that the UDP tunnel payload is an MPLS packet. As for whether the top label in the MPLS label stack is downstream-assigned or upstream-assigned, it SHOULD be determined based on the tunnel destination IP address. That is to say, if the destination IP address is a multicast address, the top label SHOULD be upstream-assigned, otherwise if the destination IP address is a unicast address, it SHOULD be downstream-assigned. Intermediate routers, upon receiving these UDP encapsulated packets, could balance these packets based on the hash of the five-tuple of UDP packets. Upon receiving these UDP encapsulated packets, the decapsulator would decapsulate them by removing the UDP headers and then process them accordingly. For other common processing procedures associated with tunneling encapsulation technologies including but not limited to Maximum Transmission Unit (MTU) and preventing fragmentation and reassembly, Time to Live (TTL) and differentiated services, the corresponding "Common Procedures" defined in [RFC4023] which are applicable for MPLS-in-IP and MPLS-in-GRE encapsulation formats SHOULD be followed.

5. Congestion Considerations

Section 3.1.3 of [RFC5405] discussed the congestion implications of UDP tunnels. As discussed in [RFC5405], because other flows can share the path with one or more UDP tunnels, congestion control [RFC2914] needs to be considered.

A major motivation for encapsulating MPLS in UDP is to improve the use of multipath (such as ECMP) in cases where traffic is to traverse routers which are able to hash on UDP Port and IP address. As such, in many cases this may reduce the occurrence of congestion and improve usage of available network capacity. However, it is also necessary to ensure that the network, including applications that use the network, responds appropriately in more difficult cases, such as when link or equipment failures have reduced the available capacity.

The impact of congestion must be considered both in terms of the effect on the rest of the network of a UDP tunnel that is consuming excessive capacity, and in terms of the effect on the flows using the UDP tunnels. The potential impact of congestion from a UDP tunnel depends upon what sort of traffic is carried over the tunnel, as well as the path of the tunnel.

MPLS is widely used to carry a wide range of traffic. In many cases MPLS is used to carry IP traffic. IP traffic is generally assumed to be congestion controlled, and thus a tunnel carrying general IP traffic (as might be expected to be carried across the Internet) generally does not need additional congestion control mechanisms. As specified in [RFC5405]:

> "IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary".

For this reason, where MPLS-in-UDP tunneling is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels MAY be used across any combination of a single service provider, multiple cooperating service providers, or across the general Internet. Internet IP traffic is generally assumed to be congestion-controlled. Similarly, in general Layer 3 VPNs are carrying IP traffic that is similarly assumed to be congestion controlled.

Whether or not Layer 2 VPN traffic is congestion controlled may depend upon the specific services being offered and what use is being made of the layer 2 services.

However, MPLS is also used in many cases to carry traffic that is not necessarily congestion controlled. For example, MPLS may be used to carry pseudowire or VPN traffic where specific bandwidth guarantees are provided to each pseudowire, or to each VPN.

In such cases service providers may avoid congestion by careful provisioning of their networks, by rate limiting of user data traffic, and/or by using MPLS Traffic Engineering (MPLS-TE) to assign specific bandwidth guarantees to each LSP. Where MPLS is carried over UDP over IP, the identity of each individual MPLS flow is in general lost and MPLS-TE cannot be used to guarantee bandwidth to specific flows (since many LSPs may be multiplexed over a single UDP tunnel, and many UDP tunnels may be mixed in an IP network).

For this reason, where the MPLS traffic is not congestion controlled, MPLS-in-UDP tunnels MUST only be used within a single service provider that utilizes careful provisioning (e.g., rate limiting at the entries of the network while over-provisioning network capacity) to ensure against congestion, or within a limited number of service providers who closely cooperate in order to jointly provide this same careful provisioning.

As such, MPLS-in-UDP MUST NOT be used over the general Internet, or over non-cooperating SPs, to carry traffic that is not congestion-controlled.

Measures SHOULD be taken to prevent non-congestion-controlled MPLS-in-UDP traffic from "escaping" to the general Internet, e.g.:

a. Physical or logical isolation of the links carrying MPLS-over-UDP from the general Internet.

b. Deployment of packet filters that block the UDP ports assigned for MPLS-over-UDP.

c. Imposition of restrictions on MPLS-in-UDP traffic by software tools used to set up MPLS-in-UDP tunnels between specific end systems (as might be used within a single data center).

d. Use of a "Managed Circuit Breaker" for the MPLS traffic as described in [I-D.fairhurst-tsvwg-circuit-breaker].

6. Security Considerations

The security problems faced with the MPLS-in-UDP tunnel are exactly the same as those faced with MPLS-in-IP and MPLS-in-GRE tunnels [RFC4023]. In other words, the MPLS-in-UDP tunnel as defined in this document by itself cannot ensure the integrity and privacy of data packets being transported through the MPLS-in-UDP tunnel and cannot enable the tunnel decapsulator to authenticate the tunnel encapsulator. In the case where any of the above security issues is concerned, the MPLS-in-UDP tunnel SHOULD be secured with IPsec or DTLS. IPsec was designed as a network security mechanism and therefore it resides at the network layer. As such, if the tunnel is secured with IPsec, the UDP header would not be visible to intermediate routers anymore in either IPsec tunnel or transport mode. As a result, the meaning of adopting the MPLS-in-UDP tunnel as an alternative to the MPLS-in-GRE or MPLS-in-IP tunnel is lost. By comparison, DTLS is better suited for application security and can better preserve network and transport layer protocol information. Specifically, if DTLS is used, the destination port of the UDP header will be filled with a value (TBD2) indicating MPLS with DTLS and the source port can still be used as an entropy field for load-sharing purposes.

If the tunnel is not secured with IPsec or DTLS, some other method should be used to ensure that packets are decapsulated and forwarded by the tunnel tail only if those packets were encapsulated by the tunnel head. If the tunnel lies entirely within a single administrative domain, address filtering at the boundaries can be used to ensure that no packet with the IP source address of a tunnel endpoint or with the IP destination address of a tunnel endpoint can enter the domain from outside. However, when the tunnel head and the tunnel tail are not in the same administrative domain, this may become difficult, and filtering based on the destination address can even become impossible if the packets must traverse the public Internet. Sometimes only source address filtering (but not destination address filtering) is done at the boundaries of an administrative domain. If this is the case, the filtering does not provide effective protection at all unless the decapsulator of an MPLS-in-UDP validates the IP source address of the packet.

This document does not require that the decapsulator validate the IP source address of the tunneled packets (with the exception that the IPv6 source address MUST be validated when UDP zero-checksum mode is used with IPv6), but it should be understood that failure to do so presupposes that there is effective destination-based (or a combination of source-based and destination-based) filtering at the boundaries. MPLS-based VPN services rely on a VPN label in the MPLS label stack to identify the VPN. Corruption of that label could leak traffic across VPN boundaries. Such leakage is highly undesirable when inter-VPN isolation is used for privacy or security reasons. When that is the case, UDP checksums SHOULD be used for MPLS-in-UDP with both IPv4 and IPv6, and in particular, UDP zero-checksum mode SHOULD NOT be used with IPv6. Each UDP checksum covers the VPN label, thereby providing increased assurance of isolation among VPNs.

7. IANA Considerations

One UDP destination port number indicating MPLS needs to be allocated by IANA:

Service Name: MPLS-in-UDP

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>.

Description: Encapsulate MPLS packets in UDP tunnels.

Reference: This document -- draft-ietf-mpls-in-udp (MPLS WG document).

Port Number: TBD1 -- To be assigned by IANA.

One UDP destination port number indicating MPLS with DTLS needs to be allocated by IANA:

Service Name: MPLS-in-UDP-with-DTLS

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>.

Description: Encapsulate MPLS packets in UDP tunnels with DTLS.

Reference: This document -- draft-ietf-mpls-in-udp (MPLS WG document).

Port Number: TBD2 -- To be assigned by IANA.

8. Contributors

Zhenbin Li

Huawei Technologies,

Beijing, China

Email: lizhenbin@huawei.com

Curtis Villamizar

Outer Cape Cod Network Consulting, LLC

Email: curtis@occnc.com

9. Acknowledgements

Thanks to Shane Amante, Dino Farinacci, Keshava A K, Ivan Pepelnjak, Eric Rosen, Andrew G. Malis, Kireeti Kompella, Marshall Eubanks, George Swallow, Loa Andersson, Vivek Kumar, Stewart Bryant, Wen Zhang, Joel M. Halpern, Noel Chiappa, Scott Brim, Alia Atlas, Alexander Vainshtein, Joel Jaeggli, Edward Crabbe, Mark Tinka, Lars Eggert, Joe Touch, Lloyd Wood, Weiguo Hao, Mark Szczesniak, Zhenxiao Liu and Xing Tong for their valuable comments and suggestions on this document.

Special thanks to Adrian Farrel for his conscientious AD review and insightful suggestion of using DTLS for securing the MPLS-in-UDP tunnels. Special thanks to Alia Atlas for her insightful suggestion of adding an applicability statement.

Thanks to Daniel King, Gregory Mirsky and Eric Osborne for their valuable MPLS-RT reviews on this document. Thanks to Charlie Kaufman for his SecDir review of this document. Thanks to Nevil Brownlee for the OPSDir review of this document. Thanks to Roni Even for the Gen-ART review of this document. Thanks to Pearl Liang for the IANA review of this documents.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

10.2. Informative References

[I-D.fairhurst-tsvwg-circuit-breaker] Fairhurst, G., "Network Transport Circuit Breakers", draft-fairhurst-tsvwg-circuit-breaker-01 (work in progress), May 2014.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, March 2005.

[RFC4817] Townsley, M., Pignataro, C., Wainner, S., Seely, T., and J. Young, "Encapsulation of MPLS over Layer 2 Tunneling Protocol Version 3", RFC 4817, March 2007.

[RFC5640] Filsfils, C., Mohapatra, P., and C. Pignataro, "Load-Balancing for Mesh Softwires", RFC 5640, August 2009.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

Authors' Addresses

Xiaohu Xu
Huawei Technologies
No.156 Beiqing Rd
Beijing 100095
CHINA

Phone: +86-10-60610041
Email: xuxiaohu@huawei.com

Nischal Sheth
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
USA

Email: nsheth@juniper.net

Lucy Yong
Huawei USA
5340 Legacy Dr
Plano, TX 75025
USA

Email: Lucy.yong@huawei.com

Carlos Pignataro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
USA

Email: cpignata@cisco.com

Yongbing Fan
China Telecom
Guangzhou
CHINA

Email: fanyb@gsta.com

Ross Callon
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rcallon@juniper.net

David Black
EMC Corporation
176 South Street
Hopkinton, MA 01748
USA

Email: david.black@emc.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2015

Tarek Saad, Ed.
Rakesh Gandhi, Ed.
Zafar Ali
Cisco Systems, Inc.
Robert H. Venator
Defense Information Systems Agency
Yuji Kamite
NTT Communications Corporation

September 30, 2014

Extensions to Resource Reservation Protocol For Re-optimization of Loosely Routed Point-to-Multipoint Traffic Engineering LSPs
draft-ietf-mpls-p2mp-loose-path-reopt-01

Abstract

For a Traffic Engineered (TE) point-to-multipoint (P2MP) Label Switched Path (LSP), it is preferable in some cases to re-evaluate and re-optimize the entire P2MP-TE LSP by re-signaling all its Source-to-Leaf (S2L) sub-LSP(s). Existing mechanisms, a mechanism for an ingress Label Switched Router (LSR) to trigger a new path re-evaluation request and a mechanism for a mid-point LSR to notify an availability of a preferred path, operate on an individual or a sub-group of S2L sub-LSP(s) basis only.

This document defines RSVP-TE signaling extensions to allow an ingress node of a P2MP-TE LSP to request the re-evaluation of the entire LSP tree containing one or more S2L sub-LSPs whose paths are loose (or abstract) hop expanded, and for a mid-point LSR to notify to the ingress node that a preferable tree exists for the entire P2MP-TE LSP. This document also defines markers to indicate beginning and end of a S2L sub-LSP descriptor list when RSVP message needs to be fragmented due to large number of S2L sub-LSPs when performing re-optimization.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [RFC2205] [RFC3209] signaling extensions for re-optimizing loosely routed Point-to-Multipoint (P2MP) Traffic Engineered (TE) Label Switched Paths (LSPs) [RFC4875] in an Multi-Protocol Label Switching (MPLS) and/or Generalized MPLS (GMPLS) networks.

A P2MP-TE LSP is comprised of one or more source-to-leaf (S2L) sub-LSPs. A loosely routed P2MP-TE S2L sub-LSP is defined as one whose path does not contain the full explicit route identifying each node along the path to the egress node at the time of its signaling by the ingress node. Such an S2L sub-LSP is signaled with no Explicit Route Object (ERO) [RFC3209], or with an ERO that contains at least one loose hop, or with an ERO that contains an abstract node that is not a simple abstract node (that is, an abstract node that identifies more than one node). This is often the case with inter-domain P2MP-TE LSPs where Path Computation Element (PCE) is not used [RFC5440].

As per [RFC4875], an ingress node may re-optimize the entire P2MP-TE LSP by re-signaling all its S2L sub-LSP(s) or may re-optimize individual or group of S2L sub-LSP(s) i.e. individual or group of destination(s).

1.1. Existing Mechanism For Re-optimizing Loosely Routed P2MP-TE LSP

[RFC4736] defines RSVP signaling extensions for re-optimizing loosely routed P2P TE LSP(s) as follows.

- A mid-point LSR that expands loose next-hop(s) sends a solicited or unsolicited PathErr with the Notify error code (25 as defined in [RFC3209]) with sub-code 6 to indicate "Preferable Path Exists" to the ingress node.

- An ingress node triggers a path re-evaluation request at all mid-point LSR(s) that expands loose next-hop(s) by setting the "Path Re-evaluation Request" flag (0x20) in SESSION_ATTRIBUTES Object in the Path message.

- The ingress node upon receiving this PathErr either solicited or unsolicited initiates re-optimization of the LSP.

[RFC4736] does not define signaling extensions specific for re-optimizing entire P2MP-TE LSP tree. Mechanisms defined in

[RFC4736] can be used for signaling the re-optimization of individual or group of S2L sub-LSP(s). However, to use [RFC4736] mechanisms for re-optimizing an entire P2MP-TE LSP tree, an ingress node needs to send the path re-evaluation requests on all (typically 100s of) S2L sub-LSPs and the mid-point LSR to notify PathErrs for all S2L sub-LSPs. Such a procedure may lead to the following issues:

- A mid-point LSR that expands loose next-hop(s) may have to accumulate the received path re-evaluation request(s) for all S2L sub-LSPs (e.g, by using a wait timer) and interpret them as a re-optimization request for the whole P2MP-TE LSP tree. Otherwise, a mid-point LSR may prematurely notify "Preferable Path Exists" for one or a sub-set of S2L sub-LSPs.

- The ingress node that receives (un)solicited PathErr notification(s) for individual S2L sub-LSP(s), may prematurely start re-optimizing the sub-set of S2L sub-LSPs. However, as mentioned in [RFC4875] Section 14.2, such sub-group based re-optimization procedure may result in data duplication that can be avoided if the entire P2MP-TE LSP tree is re-optimized using a different LSP-ID, especially if the ingress node eventually receives PathErr notifications for all S2L sub-LSPs of the P2MP-TE LSP tree.

- The ingress node may have to heuristically determine when to perform entire P2MP-TE LSP tree re-optimization versus per S2L sub-LSP re-optimization, for example, to delay re-optimization long enough to allow all PathErr(s) to be received. Once all PathErr(s) are received, the ingress node has to accumulate them to see if re-optimization of the entire P2MP-TE is necessary. Such procedures may produce undesired results due to timing related issues. This may be easily avoided by the RSVP signaling messages defined in this document.

1.2. Combining Multiple Path Messages for Re-optimization

Based on [RFC4875] (Section 14.2 "Sub-Group-Based Re-Optimization"), an ingress node may trigger path re-evaluation requests for a set of S2L sub-LSPs by combining multiple Path messages using S2L sub-LSP descriptor list. A mid-point LSR may send a PathErr message containing a list of S2L sub-LSPs transiting through the LSR to notify the ingress node. This method can alleviate the scale issue associated with sending RSVP messages for individual S2L sub-LSPs. This method is useful for re-optimizing a sub-group of S2L sub-LSPs within an LSP tree. However, this procedure can lead to following issues:

- Path message that is intended to carry the path re-evaluation request as defined in [RFC4736] with a full list of S2L sub-LSPs in S2L sub-LSPs descriptor list will be decomposed at branching LSRs, and only a subset of the S2L sub-LSPs that are routed over the same next-hop will be added in the descriptor list of the Path message propagated to downstream mid-point LSRs. Consequently, when a preferable path exists at such mid-point LSRs, the PathErr can only include the sub-set of S2L sub-LSPs traversing the LSR. \020In this case, at the ingress node there is no way to distinguish which mode of re-optimization to invoke, i.e. sub-group based re-optimization using the same LSP-ID or tree based re-optimization using a different LSP-ID.

- An LSR may fragment a large RSVP message (when a combined message may not be large enough to fit all S2L sub-LSPs). In this case, the ingress node may receive multiple PathErrs with sub-sets of S2L sub-LSPs in each (either due to the combined Path message got fragmented or combined PathErr message got fragmented) and would require additional logic to infer to re-optimize the tree (for example, waiting for some time to aggregate all possible PathErr messages before taking an action).

As discussed in Section 1.1 and Section 1.2 of this document, there is a requirement to align re-optimization of P2MP-TE LSP with P2P LSP [RFC4736] to have a mechanism to trigger re-optimization of the LSP tree by re-signaling all S2L sub-LSPs with a different LSP-ID. There is also a need to define markers to indicate beginning and end of the S2L sub-LSP descriptor list when an RSVP message is fragmented due to large number of S2L sub-LSPs in the message.

This document defines RSVP-TE signaling extensions for the ingress node of a P2MP-TE LSP to trigger the re-evaluation of the P2MP LSP tree on every hop that has a next hop defined as a loose or abstract hop for one or more S2L sub-LSP path, and a mid-point LSR to signal to the ingress node that a preferable LSP tree exists (compared to the current path) or that the whole P2MP-TE LSP must be re-optimized (because of maintenance required on the TE LSP path). This document also defines markers to indicate beginning and end of a S2L sub-LSP descriptor list when RSVP message needs to be fragmented due to large number of S2L sub-LSPs when performing re-optimization.

2. Terminology

2.1. Abbreviations

ABR: Area Border Router.

AS: Autonomous System.

ERO: Explicit Route Object.

LSR: Label Switching Router.

TE LSP: Traffic Engineering Label Switched Path.

TE LSP ingress: Head-end/source of the TE LSP.

TE LSP egress: Tail-end/destination of the TE LSP.

2.2. Nomenclatures

Domain: Routing or administrative domain such as an IGP area and an autonomous system.

Interior Gateway Protocol Area (IGP Area): OSPF Area or IS-IS level.

Inter-area TE LSP: A TE LSP whose path transits across at least two different IGP areas.

Inter-AS MPLS TE LSP: A TE LSP whose path transits across at least two different Autonomous Systems (ASes) or sub-ASes (BGP confederations).

S2L sub-LSP: Source-to-leaf sub Label Switched Path.

2.3. 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]. The reader is assumed to be familiar with the terminology in [RFC4875] and [RFC4736].

3. Signaling Procedure For Loosely Routed P2MP-TE LSP Re-optimization

3.1. Tree Based Re-optimization

To evaluate an entire P2MP-TE LSP tree on mid-point LSRs that expand loose next-hop(s), an ingress node MAY send a Path message with "P2MP-TE Tree Re-evaluation Request" defined in this document. An ingress node SHOULD select one of the S2L sub-LSPs of the P2MP-TE LSP tree transiting a mid-point LSR to trigger the re-evaluation request.

A mid-point LSR that expands loose next-hop(s) for one or more S2L sub-LSP path(s), and that receives a Path message with the "P2MP-TE Tree Re-evaluation Request" bit set, SHOULD check for a preferable P2MP-TE LSP tree by re-evaluating all S2L sub-LSP(s) that are expanded paths of the loose next-hops of the P2MP-TE LSP. If a preferable P2MP-TE LSP tree is found, the mid-point LSR MAY send an RSVP PathErr to the ingress node with Error code 25 (Notify defined in [RFC3209] and Error sub-code defined in this document "Preferable P2MP-TE Tree Exists". The mid-point LSR, in turn, SHOULD not propagate the "P2MP-TE Tree Re-evaluation Request" bit in subsequent RSVP Path messages sent downstream for the re-evaluated P2MP-TE LSP. The sending of an RSVP PathErr Notify message "Preferable P2MP-TE Tree Exists" to the ingress node SHALL notify the ingress node of the existence of a preferable P2MP-TE LSP tree. In addition, a mid-point LSR MAY send an unsolicited PathErr message with "Preferable P2MP-TE Tree Exists" PathErr code 25 to the ingress node to notify of a preferred the P2MP-TE LSP tree when it determines it exists. In this case, the mid-point LSR that expands loose next-hop(s) for one or more S2L sub-LSP path(s) SHOULD select one of the S2L sub-LSP(s) of the P2MP-TE LSP tree to send this PathErr message to the ingress node.

If no preferable tree for P2MP-TE LSP can be found, the recommended mode is that the mid-point LSR that expands loose next-hop(s) for one or more S2L sub-LSP path(s) SHOULD propagate the request downstream by setting the "P2MP-TE Tree Re-evaluation Request" bit in the LSP_ATTRIBUTES Object of RSVP Path message.

3.2. Sub-group Based Re-optimization

It might be preferable, as per [RFC4875], to re-optimize the entire P2MP-TE LSP by re-signaling all of its S2L sub-LSP(s) (Section 14.1, "Make-before-Break") or to re-optimize individual or group of S2L sub-LSP(s) i.e. individual or group of destination(s) (Section 14.2 "Sub-Group-Based Re-Optimization" in [RFC4875]), both using the same LSP-ID. For loosely routed S2L sub-LSPs, this can be achieved by using the procedures defined in [RFC4736] to re-optimize one or more S2L sub-LSP(s) of the P2MP-TE LSP.

An ingress node may trigger path re-evaluation requests for a set of S2L sub-LSPs by combining multiple Path messages using S2L sub-LSP descriptor list [RFC4875]. An S2L sub-LSP descriptor list is created using a series of S2L_SUB_LSP Objects as defined in [RFC4875]. Similarly, a mid-point LSR may send a PathErr message containing a list of S2L sub-LSPs transiting through the LSR to notify the ingress node of preferable paths available.

As per [RFC4875] (Section 5.2.3, "Transit Fragmentation of Path State Information"), when a Path message is not large enough to fit all S2L sub-LSPs in the descriptor list, an LSR may fragment the message. In this case, the LSR MAY add S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END Objects defined in this document at the beginning and at the end of the S2L sub-LSP descriptor list, respectively.

Both S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END Objects defined in this document are optional. However, a node MUST add the S2L_SUB_LSP_MARKER_END Object if it has added S2L_SUB_LSP_MARKER_BEGIN Object in the S2L sub-LSP descriptor list.

A mid-point LSR SHOULD wait to accumulate all S2L sub-LSPs before attempting to re-evaluate preferable path when a Path message for "Path Re-evaluation Request" is received with S2L_SUB_LSP_MARKER_BEGIN. An ingress node SHOULD wait to accumulate all S2L sub-LSPs before attempting to trigger re-optimization when a PathErr message with "Preferable Path Exists" is received with S2L_SUB_LSP_MARKER_BEGIN.

New objects S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END defined in this document have a wider applicability than the P2MP-TE LSP re-optimization but it is outside the scope of this document.

4. RSVP Signaling Extensions

4.1. P2MP-TE Tree Re-evaluation Request Flag

In order to trigger a tree re-evaluation request, a new flag is defined in Attributes Flags TLV of the LSP_ATTRIBUTES Object [RFC5420] as follows:

Bit Number (to be assigned by IANA): P2MP-TE Tree Re-evaluation
Request flag

The "P2MP-TE Tree Re-evaluation Request" flag is meaningful in a Path message of a P2MP-TE S2L sub-LSP and is inserted by the ingress node.

4.2. Preferable P2MP-TE Tree Exists Path Error Sub-code

In order to indicate to an ingress node that a preferable P2MP-TE LSP tree exists, the following new sub-code for PathErr code 25 (Notify Error) [RFC3209] is defined:

Sub-code (to be assigned by IANA): Preferable P2MP-TE Tree Exists sub-code

When a preferable path for P2MP-TE LSP tree exists, the mid-point LSR sends a solicited or unsolicited "Preferable P2MP-TE Tree Exists" PathErr notification to the ingress node of the P2MP-TE LSP.

4.3. Markers For S2L sub-LSP Descriptor

An S2L_SUB_LSP Object [RFC4875] identifies a particular S2L sub-LSP belonging to the P2MP-TE LSP. An S2L sub-LSP descriptor list is created using a series of S2L_SUB_LSP Objects as defined in [RFC4875].

In order to indicate the beginning and end of the S2L sub-LSP descriptor list when the RSVP message needs to be fragmented due to large number of S2L sub-LSPs, the following new types are defined for the S2L_SUB_LSP Object [RFC4875].

S2L_SUB_LSP_MARKER_BEGIN :

Class-Num 50, C-Type TBA by IANA

```
+-----------------+---------------+--------------------------+
| Length (4 bytes)| Class_Num 50  | S2L_SUB_LSP_MARKER_BEGIN |
+-----------------+---------------+--------------------------+
```

S2L_SUB_LSP_MARKER_END :

Class-Num 50, C-Type TBA by IANA

```
+-----------------+---------------+--------------------------+
| Length (4 bytes)| Class_Num 50  | S2L_SUB_LSP_MARKER_END   |
+-----------------+---------------+--------------------------+
```

The S2L_SUB_LSP_MARKER_BEGIN Object is added before adding the first S2L_SUB_LSP_IPv4 or S2L_SUB_LSP_IPv6 Object and the S2L_SUB_LSP_MARKER_END Object is added after adding the last S2L_SUB_LSP_IPv4 or S2L_SUB_LSP_IPv6 Object in the S2L sub-LSP descriptor list.

5. Compatibility

The LSP_ATTRIBUTES Object has been defined in [RFC5420] with class numbers in the form 11bbbbbb, which ensures compatibility with non-supporting nodes. Per [RFC2205], nodes not supporting this extension will ignore the new flag defined in this document but forward it without modification.

The S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END Objects have been defined with class numbers in the form 11bbbbbb, which ensures compatibility with non-supporting nodes. Per [RFC2205], nodes not supporting new S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END Objects will ignore them but forward it without modification.

6. Security Considerations

This document defines a mechanism for a mid-point LSR to notify the ingress node of a P2MP-TE LSP of the existence of a preferable tree. As per [RFC4736], in the case of a P2MP-TE LSP S2L sub-LSP spanning multiple domains, it may be desirable for a mid-point LSR to modify the RSVP PathErr message defined in this document to maintain confidentiality across different domains. Furthermore, an ingress node may decide to ignore this PathErr message coming from a mid-point LSR residing in another domain. Similarly, an mid-point LSR may decide to ignore the tree re-evaluation request originating from another ingress domain.

7. IANA Considerations

IANA is requested to administer assignment of new values for namespace defined in this document and summarized in this section.

IANA maintains a name space for RSVP-TE TE parameters "Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Parameters" (see http://www.iana.org/assignments/rsvp-te-parameters/rsvp-te-parameters.xml). From the registries in this name space "Attribute Flags", allocation of new flag is requested (Section 4.1).

IANA also maintains a name space for RSVP protocol parameters "Resource Reservation Protocol (RSVP) Parameters" (see http://www.iana.org/assignments/rsvp-parameters/rsvp-parameters.xml). From the sub-registry "Sub-Codes - 25 Notify Error" in registry "Error Codes and Globally-Defined Error Value Sub-Codes", allocation of a new error code is requested (Section 4.2). Also, from the sub-registry "Class Types or C-Types 50 S2L_SUB_LSP" in registry "Class Names, Class Numbers, and Class Types", allocation of new

C-Types is requested (Section 4.3).

7.1. P2MP-TE Tree Re-evaluation Request Flag

The following new flag is defined for the Attributes Flags TLV in the LSP_ATTRIBUTES Object [RFC5420]. The numeric value is to be assigned by IANA.

o P2MP-TE Tree Re-evaluation Request Flag:

Bit No	Attribute Flag Name	Carried in Path	Carried in Resv	Carried in RRO	Reference
TBA by IANA	P2MP-TE Tree Re-evaluation	Yes	No	No	This document

7.2. Preferable P2MP-TE Tree Exists Path Error Sub-code

As defined in [RFC3209], the Error Code 25 in the ERROR SPEC Object corresponds to a Notify Error PathErr. This document adds a new sub-code as follows for this PathErr:

o Preferable P2MP-TE Tree Exists sub-code:

Sub-code value	Sub-code Description	PathErr Code	PathErr Name	Reference
TBA by IANA	Preferable P2MP-TE Tree Exists	25	Notify Error	This document

7.3. BEGIN and END Markers For S2L sub-LSP Descriptor

As defined in [RFC4875], S2L_SUB_LSP Object is defined with Class-Number 50 to identify a particular S2L sub-LSP belonging to the P2MP-TE LSP. This document adds two new object types for this object as follows:

- S2L_SUB_LSP_MARKER_BEGIN and S2L_SUB_LSP_MARKER_END Object types:

C-Type value	Description	Reference
TBA by IANA	S2L_SUB_LSP_MARKER_BEGIN	This document
TBA by IANA	S2L_SUB_LSP_MARKER_END	This document

8. Acknowledgments

The authors would like to thank Loa Andersson, Sriganesh Kini, Curtis Villamizar, Dimitri Papadimitriou and Nobo Akiya for reviewing this document.

9. References

9.1. Normative References

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC4875] Aggarwal, R., Papadimitriou, D., and S. Yasukawa, "Extensions to Resource Reservation Protocol Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, May 2007.

[RFC5420] Farrel, A., Papadimitriou, D., Vasseur, JP., and A. Ayyangarps, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)", RFC 5420, February 2009.

9.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4736] Vasseur, JP., Ikejiri, Y. and Zhang, R, "Reoptimization of Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Loosely Routed Label Switched Path (LSP)", RFC 4736, November 2006.

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

Author's Addresses

Tarek Saad (editor)
Cisco Systems

Email: tsaad@cisco.com

Rakesh Gandhi (editor)
Cisco Systems

Email: rgandhi@cisco.com

Zafar Ali
Cisco Systems

Email: zali@cisco.com

Robert H. Venator
Defense Information Systems Agency

Email: robert.h.venator.civ@mail.mil

Yuji Kamite
NTT Communications Corporation

Email: y.kamite@ntt.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 28, 2015

S. Kini, Ed.
Ericsson
K. Kompella
Juniper
S. Sivabalan
Cisco
S. Litkowski
Orange
R. Shakir
B.T.
X. Xu
Huawei
W. Hendrickx
Alcatel-Lucent
J. Tantsura
Ericsson

October 25, 2014

Entropy labels for source routed stacked tunnels
draft-kini-mpls-spring-entropy-label-02

Abstract

Source routed tunnel stacking is a technique that can be leveraged to provide a method to steer a packet through a controlled set of segments. This can be applied to the Multi Protocol Label Switching (MPLS) data plane. Entropy label (EL) is a technique used in MPLS to improve load balancing. This document examines and describes how ELs are to be applied to source routed stacked tunnels.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The source routed stacked tunnels paradigm is leveraged by techniques such as Segment Routing (SR) [I-D.filsfils-spring-segment-routing] to steer a packet through a set of segments. This can be directly applied to the MPLS data plane, but it has implications on label stack depth.

Clarifying statements on label stack depth have been provided in [RFC7325] but they do not address the case of source routed stacked MPLS tunnels as described in [I-D.gredler-spring-mpls] or

[I-D.filsfils-spring-segment-routing] where deeper label stacks are more prevalent.

Entropy label (EL) [RFC6790] is a technique used in the MPLS data plane to provide entropy for load balancing. When using LSP hierarchies there are implications on how [RFC6790] should be applied. One such issue is addressed by [I-D.ravisingh-mpls-el-for-seamless-mpls] but that is when different levels of the hierarchy are created at different LSRs. The current document addresses the case where the hierarchy is created at a single LSR as required by source stacked tunnels.

A use-case requiring load balancing with source stacked tunnels is given in Section 3. A recommended solution is described in Section 4 keeping in consideration the limitations of implementations when applying [RFC6790] to deeper label stacks. Options that were considered to arrive at the recommended solution are documented for historical purposes in Section 5.

1.1. Requirements Language

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

Although this document is not a protocol specification, the use of this language clarifies the instructions to protocol designers producing solutions that satisfy the requirements set out in this document.

2. Abbreviations and Terminology

EL - Entropy Label

ELI - Entropy Label Identifier

ELC - Entropy Label Capability

SR - Segment Routing

ECMP - Equal Cost Multi Paths

MPLS - Multiprotocol Label Switching

SID - Segment Identifier

RLD - Readable Label Depth

OAM - Operation, Administration and Maintenance

3. Use-case for multipath load balancing in source stacked tunnels

Source stacked tunnels have several use-cases, one of which is service chaining [I-D.filsfils-spring-segment-routing-use-cases]. Consider the service-chaining network in Figure 1 that has MPLS as the data plane. The requirement of the use-case is to create a LSP from source LSR S, apply the services S1, S2 and finally terminate the LSP at destination LSR D. Local load balancing is required across the parallel links between P1 and S1. Local load balancing is also required between the ECMP paths from S1 to S2 i.e., between the paths S1-P1-P2-P3-S2 and S1-P1-P2-P4-S2. Segment routing can be used to achieve this. A segment to S1 is stacked above the segment to S2 which in turn is stacked above the segment to D. Labels for service instructions are also inserted in the stack at appropriate depths so that services S1 and S2 are executed. To achieve local load balancing the SIDs of specific interfaces is not specified. Since entropy label is a standardized [RFC6790] mechanism defined for MPLS it can be adapted to the case of source stacked tunnels. Multiple ways to apply entropy labels exist and a recommended solution is described in Section 4 and all the options considered are listed in Section 5 along with their tradeoffs. We denote SN to be the node SID of LSR N and SN{L1,L2,...} to denote the SID of the adjacency for the set for links {L1,L2,...} of LSR N and S-SvcN to denote the SID for a service at service LSR N. The label stack that the source LSR S uses for the LSP can be <SS1, S-SvcS1, SS2, S-SvcS2, SD> or <SP1, SP1{L1,L2}, S-SvcS1, SS2, S-SvcS2, SD>.

```
               +-----+               +-----+
               | S1  |        +------| P3  |------+
               +-----+        |      +-----+      |
               L1| |L2        |                   |
   +-----+     +-----+     +-----+             +-----+
   |  S  |-----| P1  |-----| P2  |             | S2  |
   +-----+     +-----+     +-----+             +-----+
                              |                   |
                              |      +-----+      |
                              +------| P4  |------+
                                     +-----+
                                        |
                                     +-----+
                                     |  D  |
                                     +-----+
```

S=Source LSR, D=Destination LSR, S1,S2=service-LSRs, L1,L2=links,
P1,P2,P3,P4=Transit LSRs

Figure 1: Service chaining use-case

4. Recommended EL solution for SPRING

The solution described in this section follows [RFC6790].

An LSR may have a limitation in its ability to read and process the label stack in order to do multipath load balancing. This limitation expressed in terms of the number of label stack entries that the LSR can read is henceforth referred to as the Readable Label Depth (RLD) capability of that LSR. If an EL does not occur within the RLD of an LSR in the label stack of the MPLS packet that it receives, then it would lead to poor load balancing at that LSR. The RLD of an LSR is a characteristic of the forwarding plane of that LSR’s implementation and determining it is outside the scope of this document.

In order for the EL to occur within the RLD of LSRs along the path corresponding to a label stack, multiple <ELI, EL> pairs MAY be inserted in the label stack as long as the labels below which they are inserted are entropy label capable. The LSR that inserts <ELI, EL> pairs MAY have limitations on the number of such pairs that it can insert and also the depth at which it can insert them. If due to any limitation, the inserted ELs are at positions such that an LSR along the path receives an MPLS packet without an EL in the label stack within that LSR’s RLD, then the load balancing performed by that LSR would be poor. Special attention should be paid when a forwarding adjacency LSP (FA-LSP) [RFC4206] is used as a link along the path of a source stacked LSP, since the labels of the FA-LSP would additionally count towards the depth of the label stack when calculating the appropriate positions to insert the ELs. The recommendations for inserting <ELI, EL> pairs are:

- An LSR that is limited in the number of <ELI, EL> pairs that it can insert SHOULD insert such pairs deeper in the stack.

- An LSR SHOULD try to insert an <ELI, EL> pair within the RLD of the maximum number of LSRs along the path as it can.

- An LSR SHOULD try to insert the minimum number of such pairs while trying to satisfy the above criteria.

A sample algorithm to insert ELs is shown below. Implementations can choose any algorithm as long as it follows the above recommendations.

```
Initialize the current EL insertion point to the
  bottommost label in the stack that is EL-capable
while local-node can push more labels OR
  top of stack has been reached {
    insert an ELI+EL at current insertion point
    move insertion point up until current EL is out of RLD
                                             AND
                                    insertion point is EL-capable
    set current insertion point to new insertion point
}
```

Figure 2: Algorithm to insert <ELI, EL> pairs in a label stack

The RLD can be advertised via protocols and those extensions would be described in a separate document.

The recommendations above are not expected to bring any additional OAM considerations beyond those described in section 6 of [RFC6790]. However, the OAM requirements and solutions for source stacked tunnels are still under discussion and future revisions of this document will address those if needed.

5. Options considered

5.1. Single EL at the bottom of the stack of tunnels

In this option a single EL is used for the entire label stack. The source LSR S encodes the entropy label (EL) below the labels of all the stacked tunnels. In Figure 1 label stack at LSR S would look like <SP1, SS1, S-SvcS1, SS2, S-SvcS2, SD, ELI, EL> <remaining packet header>. Note that the notation in [RFC6790] is used to describe the label stack. An issue with this approach is that as the label stack grows due an increase in the number of SIDs, the EL correspondingly goes deeper in the label stack. As a result, intermediate LSRs (such as P1) that have to walk the label stack at least until the EL (if found) to perform load balancing decisions have to access a larger number of bytes in the packet header when making forwarding decisions. A load balanced network design using this approach must ensure that all intermediate LSRs have the capability to traverse the maximum label stack depth in order to do effective load balancing. The use-case for which the tunnel stacking is applied would determine the maximum label stack depth.

In the case where the hardware is capable of pushing a single <ELI, EL> pair at any depth, this option is the same as the recommended solution in Section 4.

This option was discounted since there exist a number of hardware implementations which have a low maximum readable label depth. Choosing this option can lead to a loss of load-balancing using EL in a significant part of the network but that is a critical requirement in a service provider network.

5.2. An EL per tunnel in the stack

In this option each tunnel in the stack can be given its own EL. The source LSR pushes an <ELI, EL> before pushing a tunnel label when load balancing is required to direct traffic on that tunnel. For the same Figure 1 above, the source LSR S encoded label stack would be <SS1, ELI, EL1, S-SvcS1, SS2, ELI, EL2, S-SvcS2, SD> where all the ELs can have the same value. Accessing the EL at an intermediate LSR is independent of the depth of the label stack and hence independent of the specific use-case to which the stacked tunnels are applied. A drawback is that the depth of the label stack grows significantly, almost 3 times as the number of labels in the label stack. The network design should ensure that source LSRs should have the capability to push such a deep label stack. Also, the bandwidth overhead and potential MTU issues of deep label stacks should be accounted for in the network design.

In the case where the RLD is the minimum value (3) for all LSRs, all LSRs are EL capable and the LSR that is inserting <ELI, EL> pairs has no limit on how many it can insert then this option is the same as the recommended solution in Section 4.

This option was discounted due to the existence of hardware implementations that can push a limited number of labels on the label stack. Choosing this option would result in a hardware requirement to push two additional labels per tunnel label. Hence it would restrict the number of tunnels that can form a LSP and constrain the types of LSPs that can be created. This was considered unacceptable.

5.3. A re-usable EL for a stack of tunnels

In this option an LSR that terminates a tunnel re-uses the EL of the terminated tunnel for the next inner tunnel. It does this by storing the EL from the outer tunnel when that tunnel is terminated and re-inserting it below the next inner tunnel label during the label swap operation. The LSR that stacks tunnels SHOULD insert an EL below the outermost tunnel. It SHOULD NOT insert ELs for any inner tunnels. Also, the penultimate hop LSR of a segment MUST NOT pop the ELI and EL even though they are exposed as the top labels since the terminating LSR of that segment would re-use the EL for the next segment.

For the same Figure 1 above, the source LSR S encoded label stack would be <SS11, ELI, EL, S-SvcS1, SS2, S-SvcS2, SD>. At P1 the outgoing label stack would be <SS1, ELI, EL, S-SvcS1, SS2, S-SvcS2, SD> after it has load balanced to one of the links L1 or L2. At S1 the outgoing label stack would be <SS2, S-SvS2, ELI, EL, SD>. At P2 the outgoing label stack would be <SS2, ELI, EL, S-SvcS2, SD> and it would load balance to one of the nexthop LSRs P3 or P4. Accessing the EL at an intermediate LSR (e.g. P3) is independent of the depth of the label stack and hence independent of the specific use-case to which the stacked tunnels are applied.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.3.1. EL at top of stack

A slight variant of the re-usable EL option is to keep the EL at the top of the stack rather than below the tunnel label. In this case each LSR that is not terminating a segment should continue to keep the received EL at the top of the stack when forwarding the packet along the segment. An LSR that terminates a segment should use the EL from the terminated segment at the top of the stack when forwarding onto the next segment.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.4. ELs at readable label stack depths

In this option the source LSR inserts ELs for tunnels in the label stack at depths such that each LSR along the path that must load balance is able to access at least one EL. Note that the source LSR may have to insert multiple ELs in the label stack at different depths for this to work since intermediate LSRs may have differing capabilities in accessing the depth of a label stack. The label stack depth access value of intermediate LSRs must be known to create such a label stack. How this value is determined is outside the scope of this document. This value can be advertised using a protocol such as an IGP. For the same Figure 1 above, if LSR P1 needs to have the EL within a depth of 4, then the source LSR S encoded label stack would be <SS1, S-SvcS1, ELI, EL2, SS2, SD> where all the ELs would typically have the same value.

In the case where the RLD has different values along the path and the LSR that is inserting <ELI, EL> pairs has no limit on how many pairs it can insert, and it knows the appropriate positions in the stack where they should be inserted, then this option is the same as the recommended solution in Section 4.

A variant of this solution was selected which balances the number of labels that need to be pushed against the requirement for entropy.

6. Acknowledgements

The authors would like to thank John Drake and Loa Andersson for their comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document does not introduce any new security considerations beyond those already listed in [RFC6790].

9. References

9.1. Normative References

[I-D.filsfils-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-filsfils-spring-segment-routing-04 (work in progress), July 2014.

[I-D.filsfils-spring-segment-routing-use-cases]
Filsfils, C., Francois, P., Previdi, S., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., Kini, S., and E. Crabbe, "Segment Routing Use Cases", draft-filsfils-spring-segment-routing-use-cases-01 (work in progress), October 2014.

[I-D.gredler-spring-mpls]
Gredler, H., Rekhter, Y., Jalil, L., Kini, S., and X. Xu, "Supporting Source/Explicitly Routed Tunnels via Stacked LSPs", draft-gredler-spring-mpls-06 (work in progress), May 2014.

[I-D.ravisingh-mpls-el-for-seamless-mpls]
Singh, R., Shen, Y., and J. Drake, "Entropy label for seamless MPLS", draft-ravisingh-mpls-el-for-seamless-mpls-02 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC7325] Villamizar, C., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, August 2014.

9.2. Informative References

[I-D.previdi-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-previdi-isis-segment-routing-extensions-05 (work in progress), February 2014.

[I-D.psenak-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-psenak-ospf-segment-routing-extensions-05 (work in progress), June 2014.

Authors’ Addresses

Sriganesh Kini (editor)
Ericsson

Email: sriganesh.kini@ericsson.com

Kireeti Kompella
Juniper

Email: kireeti@juniper.net

Siva Sivabalan
Cisco

Email: msiva@cisco.com

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Rob Shakir
B.T.

Email: rob.shakir@bt.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Wim Hendrickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

MPLS WG K. Kompella
Internet-Draft R. Balaji
Intended status: Standards Track Juniper Networks, Inc.
Expires: April 30, 2015 G. Swallow
Cisco Systems
October 27, 2014

Label Distribution Using ARP
draft-kompella-mpls-larp-02

Abstract

This document describes extensions to the Address Resolution Protocol to distribute MPLS labels for IPv4 and IPv6 host addresses. Distribution of labels via ARP enables simple plug-and-play operation of MPLS, which is a key goal of the MPLS Fabric architecture.

Requirements Language

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

The term "server" will be used in this document to refer to an ARP/ L-ARP server; the term "host" will be used to refer to a compute server or other device acting as an ARP/L-ARP client.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes extensions to the Address Resolution Protocol (ARP) [RFC0826] to advertise label bindings for IP host addresses. While there are well-established protocols, such as LDP, RSVP and BGP, that provide robust mechanisms for label distribution, these protocols tend to be relatively complex, and often require detailed configuration for proper operation. There are situations where a simpler protocol may be more suitable from an operational standpoint. An example is the case where an MPLS Fabric is the underlay technology in a Data Centre; here, MPLS tunnels originate from host machines. The host thus needs a mechanism to acquire label bindings to participate in the MPLS Fabric, but in a simple, plug-and-play manner. Existing signaling/routing protocols do not always meet this need. Labeled ARP (L-ARP) is a proposal to fill that gap.

[TODO-MPLS-FABRIC] describes the motivation for using MPLS as the fabric technology.

1.1. Approach

ARP is a nearly ubiquitous protocol; every device with an Ethernet interface, from hand-helds to hosts, have an implementation of ARP. ARP is plug-and-play; ARP clients do not need configuration to use ARP. That suggests that ARP may be a good fit for devices that want to source and sink MPLS tunnels, but do so in a zero-config, plug-and-play manner, with minimal impact to their code.

The approach taken here is to create a minor variant of the ARP protocol, labeled ARP (L-ARP), which is distinguished by a new hardware type, MPLS-over-Ethernet. Regular (Ethernet) ARP (E-ARP) and L-ARP can coexist; a device, as an ARP client, can choose to send out an E-ARP or an L-ARP request, depending on whether it needs Ethernet or MPLS connectivity. Another device may choose to function as an E-ARP server and/or an L-ARP server, depending on its ability to provide an IP-to-Ethernet and/or IP-to-MPLS mapping.

2. Overview of Ethernet ARP

In the most straightforward mode of operation [RFC0826], ARP queries are sent to resolve "directly connected" IP addresses. The ARP query is broadcast, with the Target Protocol Address field (see Section 5 for a description of the fields in an ARP message) carrying the IP address of another node in the same subnet. All the nodes in the LAN receive this ARP query. All the nodes, except the node that owns the IP address, ignore the ARP query. The IP address owner learns the MAC address of the sender from the Source Hardware Address field in the ARP request, and unicasts an ARP reply to the sender. The ARP reply carries the replying node’s MAC address in the Source Hardware Address field, thus enabling two-way communication between the two nodes.

A variation of this scheme, known as "proxy ARP" [RFC2002], allows a node to respond to an ARP request with its own MAC address, even when the responding node does not own the requested IP address. Generally, the proxy ARP response is generated by routers to attract traffic for prefixes they can forward packets to. This scheme requires the host to send ARP queries for the IP address the host is trying to reach, rather than the IP address of the router. When there is more than one router connected to a network, proxy ARP enables a host to automatically select an exit router without running any routing protocol to determine IP reachability. Unlike regular ARP, a proxy ARP request can elicit multiple responses, e.g., when more than one router has connectivity to the address being resolved. The sender must be prepared to select one of the responding routers.

Yet another variation of the ARP protocol, called ’Gratuitous ARP’ [RFC2002], allows a node to update the ARP cache of other nodes in an unsolicited fashion. Gratuitous ARP is sent as either an ARP request or an ARP reply. In either case, the Source Protocol Address and Target Protocol Address contain the sender’s address, and the Source Hardware Address is set to the sender’s hardware address. In case of a gratuitous ARP reply, the Target Hardware Address is also set to the sender’s address.

3. L-ARP Protocol Operation

The L-ARP protocol builds on the proxy ARP model, and also leverages gratuitous ARP model for asynchronous updates.

In this memo, we will refer to L-ARP clients (that make L-ARP requests) and L-ARP servers (that send L-ARP responses). In Figure 1, H1, H2 and H3 are L-ARP clients, and T1, T2 and T3 are L-ARP servers. T is a member of the MPLS Fabric that may not be an L-ARP server. Within the MPLS Fabric, the usual MPLS protocols (IGP, LDP, RSVP-TE) are run. Say H1, H2 and H3 want to establish MPLS tunnels to each other (for example, they are using BGP MPLS VPNs as the overlay virtual network technology). H1 might also want to talk to a member of the MPLS Fabric, say T.

```
                                . . . . . .
                               .           .
                      H1 --- T1             T4
                         \   .     MPLS      .
                          \  .               .
                           \ .    Fabric     .
                      H2 --- T2             T3 --- H3
                              .            .
                               . . . . . .
```

Figure 1

3.1. Basic Operation

A node (say H1) that needs an MPLS tunnel to a destination (say H3) broadcasts over all its interfaces an L-ARP query with the Target Protocol Address set to H3. A node that has reachability to H3 (such as T1 or T2) sends an L-ARP reply with the Source Hardware Address set to a locally-allocated MPLS label plus its Ethernet MAC address.

After receiving one or more L-ARP replies, H1 can select either T1 or T2 to send MPLS packets that are destined to H3. As described later, the L-ARP response may contain certain parameters that enable the client to make an informed choice of the routers.

As with standard ARP, the validity of the MPLS label obtained using L-ARP is time-bound. The client should periodically resend its L-ARP requests to obtain the latest information, and time out entries in its ARP cache if such an update is not forthcoming. Once an L-ARP server has advertised a label binding, it MUST NOT change the binding until expiry of the binding's validity time.

The mechanism defined here is simplistic; see Section 4.

3.2. Asynchronous operation

The preceding sections described a request-response based model. In some cases, the L-ARP server may want to asynchronously update its clients. L-ARP uses the gratuitous ARP model [RFC2002] to "push" such changes.

In a pure "push" model, a device may send out updates for all prefixes it knows about. This naive approach will not scale well. This memo specifies a mode of operation that is somewhere between "push" and "pull" model. An L-ARP server does not advertise any binding for a prefix until at least one L-ARP client expresses interest in that prefix (by initiating an L-ARP query). As long as the server has at least one interested client for a prefix, the server sends unsolicited (aka gratuitous, though the term is less appropriate in this context) L-ARP replies when a prefix's reachability changes. The server will deem the client's interest in a prefix to have ceased when it does not hear any L-ARP queries for some configured timeout period.

3.3. Client-Server Synchronization

In an L-ARP reply, the server communicates several pieces of information to the client: its hardware address, the MPLS label, Entropy Label capability and metric. Since ARP is a stateless protocol, it is possible that one of these changes without the client knowing, which leads to a loss of synchronization between the client and the server. This loss of synchronization can have several bad effects

If the server's hardware address changes or the MPLS label is repurposed by the server for a different purpose, then packets may be sent to the wrong destination. The consequences can range from suboptimally routed packets to dropped packets to packets being delivered to the wrong customer, which may be a security breach. This last may be the most troublesome consequence of loss of synchronization.

If a destination transitions from entropy label capable to entropy label incapable (an unlikely event) without the client knowing, then packets encapsulated with entropy labels will be dropped. A transition in the other direction is relatively benign.

If the metric changes without the client knowing, packets may be suboptimally routed. This may be the most benign consequence of loss of synchronization.

3.4. Applicability

L-ARP can be used between a host and its Top-of-Rack switch in a Data Center. L-ARP can also be used between a DSLAM and its aggregation switch going to the B-RAS. More generally, L-ARP can be used between an "access node" and its first hop MPLS-enabled device in the context of Seamless MPLS [reference]. In all these cases, L-ARP can handle the presence of multiple connections between the access device and its first hop devices.

ARP is not a routing protocol. The use of L-ARP should be limited to cases where the L-ARP client has a small number of one-hop connections to L-ARP servers. The presence of a complex topology between the L-ARP client and server suggests the use of a different protocol.

3.5. Backward Compatibility

Since L-ARP uses a new hardware type, it is backward compatible with "regular" ARP. ARP servers and clients MUST be able to send out, receive and process ARP messages based on hardware type. They MAY choose to ignore requests and replies of some hardware types; they MAY choose to log errors if they encounter hardware types they do not recognize; however, they MUST handle all hardware types gracefully. For hardware types that they do understand, ARP servers and clients MUST handle operation codes gracefully, processing those they understand, and ignoring (and possibly logging) others.

4. For Future Study

The L-ARP specification is quite simple, and the goal is to keep it that way. However, inevitably, there will be questions and features that will be requested. Some of these are:

1. Keeping L-ARP clients and servers in sync. In particular, dealing with:

    A. client and/or server restart

    B. lost packets

    C. timeouts

2. Withdrawing a response.

3. Dealing with scale.

4. If there are many servers, which one to pick?

5. How can a client make best use of underlying ECMP paths?

6. and probably many more.

In all of these, it is important to realize that, whenever possible, a solution that places most of the burden on the server rather than on the client is preferable.

5. L-ARP Message Format

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ar$hrd              |           ar$pro              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ar$hln    |    ar$pln     |           ar$op               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$sha (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$spa (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$tha (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$tpa (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$lst (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      ar$att (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: L-ARP Packet Format

ar$hrd Hardware Type: MPLS-over-Ethernet. The value of the field used here is [HTYPE-MPLS-TBD]. To start with, we will use the experimental value HW_EXP2 (256)

ar$pro Protocol Type: IPv4/IPv6. The value of the field used here is 0x0800 to resolve an IPv4 address and 0x86DD to resolve an IPv6 address.

ar$hln Hardware Length: the value of the field used here is 6.

ar$pln Protocol Address Length: for an IPv4 address, the value is 4; for an IPv6 address, it is 16.

ar$op Operation Code: set to 1 for request, 2 for reply, and 10 for ARP-NAK. Other op codes may be used, but this is not anticipated at this time.

ar$sha Source Hardware Address: In an L-ARP message, Source Hardware Address is the 6 octets of the sender’s MAC address.

ar$spa Source Protocol Address: In an L-ARP message, this field carries the sender’s IP address.

ar$tha Target Hardware Address: In an L-ARP query message, Target Hardware Address is the all-ones Broadcast MAC address; in an L-ARP reply message, it is the client’s MAC address.

ar$tpa Target Protocol Address: In an L-ARP message, this field carries the IP address for which the client is seeking an MPLS label.

ar$lst Label Stack: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries the MPLS label stack in the format below.

ar$att Attribute TLV: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries attributes for the MPLS label stack in the format below.

Figure 3 describes the format of MPLS Label Stack carried in L-ARP. Figure 4 describes the format of Attribute TLV carried in L-ARP.

```
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |    # labels   |         MPLS Label (20 bits)         |E|Z|Z|Z|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |         MPLS Label (20 bits)         |E|Z|Z|Z| ...           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: MPLS Label Stack Format

MPLS Label Stack: This field contains the MPLS label stack for the client to use to get to the target. Each label is 3 octets; the Length is 3*(number of labels). This field is valid only in an L-ARP request message.

E-bit: Entropy Capability

This field indicates whether the label stack of MPLS data packets sent with the label in this advertisement can contain Entropy Label or not. If this flag is set, the client has the option of inserting ELI and EL as specified in [RFC6790]. The client can choose not to insert ELI/EL pair, if it does not support Entropy Labels, or the local policy does not permit the client to insert ELI/EL. If this flag is clear, the client must not insert ELI/EL into the label stack when sending packets with the advertised L-ARP label.

Z These bits are not used, and SHOULD be set to zero on sending and ignored on receipt.

If other parameters are deemed useful in the L-ARP reply, they will be added as needed.

```
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |      Type     |    Length     |     Metric (4 octets) ...     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  ...  Metric                  | ...                           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Attribute TLV

6. Security Considerations

TODO

7. IANA Considerations

TODO

8. Acknowledgments

Many thanks to Shane Amante for his detailed comments and suggestions. Many thanks to the team in Juniper prototyping this work for their suggestions on making this variant workable in the context of existing ARP implementations. Thanks too to Luyuan Fang, Alex Semenyaka and Dmitry Afanasiev for their comments and encouragement.

9. Normative References

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC2002] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

Authors’ Addresses

Kireeti Kompella
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

Balaji Rajagopalan
Juniper Networks, Inc.
Prestige Electra, Exora Business Park
Marathahalli - Sarjapur Outer Ring Road
Bangalore 560103
India

Email: balajir@juniper.net

George Swallow
Cisco Systems
1414 Massachusetts Ave
Boxborough, MA 01719
US

Email: swallow@cisco.com

MPLS WG   K. Kompella
Internet-Draft   Juniper Networks, Inc.
Intended status: Standards Track   October 26, 2014
Expires: April 29, 2015

Resilient MPLS Rings
draft-kompella-mpls-rmr-00

Abstract

This document describes the use of the MPLS control and data planes on ring topologies. It describes the special nature of rings, and proceeds to show how MPLS can be effectively used in such topologies. It describes how MPLS rings are configured, auto-discovered and signaled, as well as how the data plane works. Companion documents describe the details of discovery and signaling for specific protocols.

Requirements Language

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

Status of this Memo

This Internet-Draft will expire on April 29, 2015.

Copyright Notice

Table of Contents

1. Introduction

Rings are a very common topology in transport networks. A ring is the simplest topology offering link and node resilience. Rings are nearly ubiquitous in access and aggregation networks. As MPLS increases its presence in such networks, and takes on a greater role in transport, it is imperative that MPLS handles rings well; this is not the case today.

This document describes the special nature of rings, and the special needs of MPLS on rings. It then shows how these needs can be met in several ways, some of which involve extensions to protocols such as IS-IS [RFC1195], OSPF[RFC2328], RSVP-TE [RFC3209] and LDP [RFC5036].

1.1. Definitions

A (directed) graph G = (V, E) consists of a set of vertices (or nodes) V and a set of edges (or links) E. An edge is an ordered pair of nodes (a, b), where a and b are in V. (In this document, the terms node and link will be used instead of vertex and edge.)

A ring is a subgraph of G. A ring consists of a subset of nodes {R_i, 1 <= i <= n} of V. For convenience, we define R_0 = R_n. The edges {(R_i, R_i+1) and (R_i+1, R_i), 0 <= i < n} must be a subset of E. We define the direction from node R_i to R_i+1 (0 <= i < n) (and hence from R_n = R_0 to R_1) as "downstream" (DS) and the reverse direction as "upstream" (US). As there may be several rings in a graph, we number each ring with a distinct "Ring ID" RID.

```
                               R1 . . . R2
                             .             .
                          R8                 R3
                       |  .        Ring       .  |
              Upstream |  .                   .  | Downstream
                       v  .      RID = 17     .  v
                          R7                 R4
                             .             .
                               R6 . . . R5
```

Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

RL_k: A Ring LSP anchored on node R_k is denoted RL_k.

DL_jk (UL_jk): A label allocated by R_j for RL_k in the DS (US) direction.

P_jk (Q_jk): A Path (Resv) message sent by R_j for RL_k.

2. Motivation

A ring is the simplest topology that offers resilience. This is perhaps the main reason to lay out fiber in a ring. Thus, effective mechanisms for fast failover on rings are needed. Furthermore, there are large numbers of rings. Thus, configuration of rings needs to be as simple as possible. Finally, bandwidth management on access rings is very important, as bandwidth is generally quite constrained here.

The goals of this document are to present mechanisms for improved MPLS-based resilience in ring networks (using ideas that are reminiscent of Bidirectional Line Switched Rings), automatic bring-up of LSPs, better bandwidth management and auto-hierarchy. These goals can be achieved using extensions to existing IGP and MPLS signaling protocols, using central provisioning, or in other ways.

3. Theory of Operation

Say a ring has n nodes R_i, 0 <= i <= n, where R_0 = R_n. Each node is the anchor of one or more Ring LSPs. A Ring LSP RL_i anchored on node R_i consists of two counter-rotating LSPs that start and end at R_i. A ring LSP is "multipoint"; any node R_j can use RL_i to send traffic to R_i in either direction. Bidirectional connectivity between nodes R_i and R_j is achieved by using ring LSPs RL_j (to reach R_j) and RL_i (to reach R_i); each can be used in either direction.

3.1. Configuration

An MPLS ring is configured by assigning RIDs to all the nodes in the ring. The links between adjacent ring nodes are ring links (unless told otherwise); this may also be configured, or it may be discovered, say by means of IGP hellos. Once ring nodes and ring links are identified, the ring has been defined.

Ring LSPs are not provisioned; they are created automatically when an MPLS ring is defined. Each node R_j allocates DS and US labels for each ring LSP RL_k and sends these to its ring neighbors. The signaling protocol used to send labels can be RSVP-TE or LDP; these extensions will be described later. When R_j receives DS and US labels for RL_k, it can install LFIB entries for RL_k.

3.2. Auto-discovery

A link-state IGP such as IS-IS or OSPF can be used to simplify the configuration of MPLS rings. Details will be given in a companion document.

3.3. Signaling

Both RSVP-TE and LDP, with appropriate extensions, can be used to signal ring LSPs. Details will be given in companion documents.

3.4. Installing Primary LFIB Entries

In setting up RL_k, a node R_j sends out two labels: DL_jk to R_j-1 and UL_jk to R_j+1. R_j also receives two labels: DL_j+1,k from R_j+1, and UL_j-1,k from R_j-1. R_j can now set up the forwarding entries for RL_k. In the DS direction, R_j swaps incoming label DL_jk with DL_j+1,k with next hop R_j+1. In the US direction, R_j swaps incoming label UL_jk with UL_j-1,k with next hop R_j-1. R_k does not install LFIB entries in this manner.

3.5. Installing FRR LFIB Entries

At the same time that R_j sets up its DS and US LFIB entries, it can also set up the protection forwarding entries for RL_k. In the DS direction, R_j sets up an FRR LFIB entry to swap incoming label DL_jk with UL_j-1,k with next hop R_j-1. In the US direction, R_j sets up an FRR LFIB entry to swap incoming label UL_jk with DL_j+1,k with next hop R_j+1. Again, R_k does not install FRR LFIB entries in this manner.

3.6. Protection

Note that in this scheme, there are no protection LSPs as such -- no node or link bypasses, nor detours, nor LFA-type protection. Protection is via the "other" direction around the ring, which is why ring LSPs are in counter-rotating pairs.

If a node R_j detects a failure DS from R_j+1, it switches traffic on all DS ring LSPs to the US direction using the FRR LFIB entries. This switchover can be very fast, as the FRR LFIB entries can be preprogrammed. If the detection is fast too, then traffic loss is minimal.

R_j then sends an indication to R_j-1 that the DS direction is not working, so that R_j-1 can similarly switch traffic to the US direction. These indications propagate US until each traffic source on the ring uses the US direction. Thus, within a short period, traffic will be flowing in the optimal path, given that there is a failure on the ring. This contrasts with (say) bypass protection, where until the ingress recomputes a new path, traffic will be suboptimal.

4. Security Considerations

It is not anticipated that either the notion of MPLS rings or the extensions to various protocols to support them will cause new security loopholes. As this document is updated, this section will also be updated.

5. IANA Considerations

There are no requests to IANA for this document.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

Author's Address

Kireeti Kompella
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

G. Mirsky
J. Tantsura
Ericsson
I. Varlashkin
Google

October 23, 2014

Bidirectional Forwarding Detection (BFD) Directed Return Path
draft-mirsky-mpls-bfd-directed-01

Abstract

Bidirectional Forwarding Detection (BFD) is expected to monitor bi-directional paths. When a BFD session monitors in its forward direction an explicitly routed path there is a need to be able to direct far-end BFD peer to use specific path as reverse direction of the BFD session.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The [RFC5880], [RFC5881], and the [RFC5883] established BFD protocol for IP networks and the [RFC5884] set rules of using BFD Asynchronous mode over IP/MPLS LSPs. All standards implicitly assume that the far-end BFD peer will use the best route regardless of route being used to send BFD control packets towards it. As result, if the near-end BFD peer sends its BFD control packets over explicit path that is diverging from the best route, then reverse direction of the BFD session is likely not to be on co-routed bi-directional path with the forward direction of the BFD session. And because BFD control packets are not guaranteed to cross the same links and nodes in both directions detection of Loss of Continuity (LoC) defect in forward direction is not guaranteed or is free of positive negatives.

This document proposes to use BFD Return Path TLV extension to LSP Ping [RFC4379] to instruct the far-end BFD peer to use explicit path for its BFD control packets associated with the particular BFD session. As a special case, forward and reverse directions of the BFD session can form bi-directional co-routed associated channel.

1.1. Conventions used in this document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

MPLS: Multiprotocol Label Switching

LSP: Label Switching Path

LoC: Loss of Continuity

1.1.2. Requirements Language

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

2. Problem Satement

BFD is best suited to monitor bi-directional co-routed paths. In most cases, in IP and IP/MPLS networks the best route between two IP nodes is likely to be co-routed in the stable network environment so that implicit BF D requirement is being fulfilled. If BFD is tasked to monitor unidirectional explicitly routed path, e.g. MPLS LSP, its control packets in forward direction would be in-band due to mechanism defined in [RFC5884] and [RFC5586]. But the reverse direction of the BFD session would still follow the best route and that presents following problems in regard to detecting defects on the unidirectional explicit path:

- failure detection on the reverse path cannot be interpreted as bi-directional failure and thus trigger, for example, protection switchover of the forward direction;

- if reverse direction is in Down state, the head-end node would not receive indication of forward direction failure from its far-end peer.

To address these challenges the far-end BFD peer should be instructed to use specific path for its control packets.

3. Direct Reverse BFD Path

3.1. Case of MPLS Data Plane

LSP ping, defined in [RFC4379], uses BFD Discriminator TLV [RFC5884] to bootstrap a BFD session over an MPLS LSP. This document defines a new TLV, BFD Reverse Path TLV, that MUST contain a single sub-TLV that can be used to carry information about reverse path for the specified in BFD Discriminator TLV session.

3.1.1. BFD Reverse Path TLV

The BFD Reverse Path TLV is an optional TLV within the LSP ping protocol. However, if used, the BFD Discriminator TLV MUST be included in an Echo Request message as well. If the BFD Discriminator TLV is not present when the BFD Reverse Path TLV is included, then it MUST be treated as malformed Echo Request, as described in [RFC4379].

The BFD Reverse Path TLV carries the specified path that BFD control packets of the BFD session referenced in the BFD Discriminator TLV are required to follow. The format of the BFD Reverse Path TLV is as presented in Figure 1.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   BFD Reverse Path TLV Type   |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Reverse Path                           |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: BFD Reverse Path TLV

BFD Reverse Path TLV Type is 2 octets in length and value to be assigned by IANA.

Length is 2 octets in lengh and defines the length in octets of the Reverse Path field.

Reverse Path field contains a sub-TLV. Any Target FEC sub-TLV, already or in the future defined, from IANA sub-registry Sub-TLVs for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be used in this field. Only one sub-TLV MUST be included in the Reverse Path TLV. If more then one sub-TLVs are present in the Reverse Path TLV, then only the first sub-TLV MUST be used and the rest MUST be silently discarded.

3.1.2. Segment Routing Tunnel sub-TLV

With MPLS data plane explicit path can be either Static or RSVP-TE LSP, or Segment Routing tunnel. In case of Static or RSVP-TE LSP [RFC7110] defined sub-TLVs to identify explicit return path. For the Segment Routing with MPLS data plane case a new sub-TLV is defined in this document as presented in Figure 2.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  SegRouting MPLS sub-TLV Type |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Label Stack Element                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Label Stack Element                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Segment Routing MPLS Tunnel sub-TLV

The Segment Routing Tunnel sub-TLV Type is two octets in length, and will be allocated by IANA.

The Segment Routing Tunnel sub-TLV MAY be used in Reply Path TLV defined in [RFC7110]

3.2. Case of IPv6 Data Plane

IPv6 can be data plane of choice for Segment Routed tunnels [I-D.previdi-6man-segment-routing-header]. In such networks the BFD Reverse Path TLV described in Section 3.1.1 can be used as well. IP networks, unlike IP/MPLS, do not require use of LSP ping with BFD Discriminator TLV[RFC4379] to bootstrap BFD session. But to specify reverse path of a BFD session in IPv6 environment the BFD Discriminator TLV MUST be used along with the BFD Reverse Path TLV. The BFD Reverse Path TLV in IPv6 network MUST include sub-TLV.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  SegRouting IPv6 sub-TLV Type |          Length               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                          IPv6 Prefix                          |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                          IPv6 Prefix                          |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Segment Routing IPv6 Tunnel sub-TLV

3.3. Bootstrapping BFD session with BFD Reverse Path over Segment Routed tunnel

As discussed in [I-D.kumarkini-mpls-spring-lsp-ping] introduction of Segment Routing network domains with MPLS dataplane adds three new sub-TLVs that may be used with Target FEC TLV. Section 6.1 addresses use of new sub-TLVs in Target FEC TLV in LSP ping and LSP traceroute. For the case of LSP ping the [I-D.kumarkini-mpls-spring-lsp-ping] states that:

"Initiator MUST include FEC(s) corresponding to the destination segment.

Initiator MAY include FECs corresponding to some or all of segments imposed in the label stack by the initiator to communicate the segments traversed. "

When LSP ping is used to bootstrap BFD session this document updates this and defines that LSP Ping MUST include the FEC corresponding to the destination segment and SHOULD NOT include FECs corresponding to some or all of segment imposed by the initiator. Operationally such restriction would not cause any problem or uncertainty as LSP ping with FECs corresponding to some or all segments or traceroute may preceed the LSP ping that bootstraps the BFD session.

4. IANA Considerations

4.1. TLV

The IANA is requested to assign a new value for BFD Reverse Path TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "TLVs and sub-TLVs" sub-registry.

Value	Description	Reference
X (TBD1)	BFD Reverse Path TLV	This document

Table 1: New BFD Reverse Type TLV

4.2. Sub-TLV

The IANA is requested to assign one new sub-TLV type from "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "Sub-TLVs for TLV Types 1, 16, and 21" sub-registry.

Value	Description	Reference
X (TBD2)	Segment Routing MPLS Tunnel sub-TLV	This document
X (TBD3)	Segment Routing IPv6 Tunnel sub-TLV	This document

Table 2: New Segment Routing Tunnel sub-TLV

5. Security Considerations

Sequirity considerations discussed in [RFC5880], [RFC5884], and [RFC4379], apply to this document.

6. Acknowledgements

7. Normative References

[I-D.kumarkini-mpls-spring-lsp-ping]
Kumar, N., Swallow, G., Pignataro, C., Akiya, N., Kini, S., Gredler, H., and M. Chen, "Label Switched Path (LSP) Ping/Trace for Segment Routing Networks Using MPLS Dataplane", draft-kumarkini-mpls-spring-lsp-ping-01 (work in progress), July 2014.

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., and I. Leung, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-02 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

[RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, January 2014.

Authors’ Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Network Working Group L. Zhang
Internet-Draft L. Zheng
Intended status: Standards Track S. Aldrin
Expires: March 29, 2015 Huawei Technologies
September 25, 2014

YANG Data Model for MPLS-TP Operations, Administration, and Maintenance (OAM)
draft-zhang-mpls-tp-yang-oam-00

Abstract

This document presents the YANG Data model for MPLS-TP OAM, including the basic functions of Fault Management and Performance Monitoring.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document presents the YANG Data model for MPLS-TP OAM, including the basic functions of Fault Management and Performance Monitoring.

2. Conventions used in this document

2.1. Terminology

CC - Continuity Check

CV - Conectivity Verification

LM - Loss Measurement

ME - Maintenance Entity

MEG - Maintenance Entity Group

MEP - Maintenance Entity Group End Point

MIP - Maintenance Entity Group Intermediate Point

PM - Performance Monitoring

PW - Pseudowire

DM - Packet Delay Measurement

AIS - Alarm Indication Signal

LKR - Lock Report

3. Design of the Data Model

At the top of the Model is the global configuration, which indicate the MPLS-TP OAM basic information.

Under the global configuration is Maintenance Entity Group. Each Maintenance Entity Group is associated with a Maintenance Entity Group Name and a Maintenance Entity Group level, and a certain type of Maintenace Entity, such as PW, TE, SECTION and so on.

Under each MEG, there can be one or more MEs (Maintenance Entity), which should be consistent with the certain ME type assigned above. Each ME has its own indication need to be configurated exactly.

Subsequently, the basic function of Fault Management and Performance Monitoring for each MEG should be offered. In order to facilitate zero-touch experience, this document defines a default value of the related detect parameters, such as detection intervals, the exp of OAM packet and OAM packet size.

3.1. MPLS-TP OAM Global Configuration

The container "global" is the top level container. Within the container "global", separate leaf nodes are maintained.

```
   +--rw global
      |  +--rw aisEnable?     Enable
      .
      .
```

Figure 1 Snippet of data hierarchy related to MPLS-TP OAM Global

3.2. Maintenance Entity Group (MEG) Configuration

The container "megs" is the second level container within the MPLS-TP oam module. Within the container "MEGs", separate lists are maintained per MEG. The MEG list uses the key MEG-name for indexing.

```
   +--rw global
      |  +--rw aisEnable?     Enable
      +--rw megs
      |  +--rw meg* [megName]
      |     +--rw megName           string
      |     +--rw meType?           mplstpoamMeType
      |     +--rw megId?            string
      |     +--rw megLevel?         uint8
      |     +--rw oamActiveState?   mplstpoamActiveType

      .
      .
```

Figure 2 Snippet of data hierarchy related to MPLS-TP OAM MEGs

3.3. Maintenance Entities (MEs) Configuration

Within a given Maintenance Entity Group there can be one or more Maintenance Entity (ME), which should be consistent with the certain ME type assigned within megs . Different types of MEs are represented as different list and indexed by theire own key.

3.3.1. Pseudo Wire (PW) Configuration

```
+--rw global
   |  .
   |  .
   +--rw megs
   |  +--rw meg* [megName]
   |     .
   |     .
   |     +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         |  +--rw peerIp                  inet:ip-address
         |  +--rw vcId                    uint32
         |  +--rw vcType                  mplstpoamVctype
         |  +--rw remotePeerIp            inet:ip-address
         |  +--rw remoteVcId              mplstpoamVctype
         |  +--rw remoteVcType            mplstpoamVctype
         |  +--rw mepId?                  uint16
         |  +--rw remoteMepId?            uint16
         |  +--rw vllttl?                 uint8
   .
   .
```

Figure 3 Snippet of data hierarchy related to ME of PW

3.3.2. Traffic Engineering (TE) Configuration

```
+--rw global
   .
   .
   +--rw megs
   |  +--rw meg* [megName]
         .
         .
      +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
        .
        .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         |  +--rw tunnelName              string
         |  +--rw tunnelId                uint32
         |  +--rw ingressLsrId            inet:ip-address
         |  +--rw mepId?                  uint16
         |  +--rw remoteMepId?            uint16
         |  +--rw reverseTunnelName       string
         |  +--rw reverseTunnelId?        uint16
         |  +--rw reverseIngrLsrId?       inet:ip-address

   .
   .
```

Figure 4 Snippet of data hierarchy related to ME of TE

3.3.3. Section Configuration

```
+--rw global
   .
   .
   +--rw megs
   |  +--rw meg* [megName]
         .
         .
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         .
         .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         .
         .
         +--rw section* [sectionId]
         |  +--rw sectionId                uint64
         |  +--rw ifName?                  string
         |  +--rw peerIp                   inet:ip-address
         |  +--rw peerLsrId?               inet:ip-address
         |  +--rw mepId?                   uint16
         |  +--rw remoteMepId?             uint16
   .
   .
```

Figure 5 Snippet of data hierarchy related to ME of Section

3.3.4. Virtual Leased Line (VLL) Maintenance Entity Group Intermediate Points(MIPs) Configuration

```
+--rw global
   .
   .
   +--rw megs
   |  +--rw meg* [megName]
         .
         .
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         .
         .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         .
         .
         +--rw section* [sectionId]
         .
         .
         +--rw vllMip* [peerIp vcId switchPeerIp switchVcId vcType instanceName]

         |  +--rw peerIp          inet:ip-address
         |  +--rw vcId            uint32
         |  +--rw switchPeerIp    inet:ip-address
         |  +--rw switchVcId      uint32
         |  +--rw vcType          mplstpoamVctype
         |  +--rw instanceName    string
   .
   .
```

Figure 6 Snippet of data hierarchy related to ME of VLL-MIP

3.3.5. Traffic Engineering (TE) Maintenance Entity Group Intermediate Points(MIPs) Configuration

```
+--rw global
   .
   .
   +--rw megs
   |  +--rw meg* [megName]
         .
         .
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         .
         .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         .
         .
         +--rw section* [sectionId]
         .
         .
         +--rw vllMip* [peerIp vcId switchPeerIp switchVcId vcType instanceName]
         .
         .
         +--rw teMip* [lspName]
         |  +--rw lspName        string
   .
   .
```

Figure 7 Snippet of data hierarchy related to ME of TE-MIP

3.4. MPLS-TP OAM Fault Management And Performance Moinitoring Configuration

3.4.1. Fault Management Configuration

The container "cc", "cv", "ais" and "lkr" are indicate the Fault Management tools of "Continuity Check","Connectivity Verification". "Alarm Indication Signal" and "Lock Report", which are used by the MEs of MEG.

```
+--rw meg* [megName]
         .
         .
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         .
         .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         .
         .
         +--rw section* [sectionId]
         .
         .
         +--rw vllMip* [peerIp vcId switchPeerIp switchVcId vcType instanceName]
         .
         .
         +--rw teMip* [lspName]
         .
         .
         +--rw cc
         |  +--rw ccSessionMode?            mplstpoamCcSessionMode
         |  +--rw ccAuthenticationEnable?   Enable
         |  +--rw ccExp?                    uint8
         |  +--rw ccTransmitInterval?       mplstpoamCcInterval
         |  +--rw ccRecieveInterval?        mplstpoamCcInterval
         |  +--rw ccDetectMultiplier?       mplstpoamCcDetectMultiplier
         |  +--rw ccEnable?                 Enable
         +--rw cv
         |  +--rw cvSessionMode?            mplstpoamCcSessionMode
         |  +--rw cvAuthenticationEnable?   Enable
         |  +--rw cvExp?                    uint8
         |  +--rw cvInterval?               mplstpoamCvInterval
         |  +--rw cvDetectMultiplier?       mplstpoamCvDetectMultiplier
         |  +--rw cvEnable?                 Enable
         +--rw ais
         |  +--rw aisExp?        uint8
         |  +--rw aisInterval?   mplstpoamAisInterval
         +--rw lkr
         |  +--rw lkrExp?        uint8
         |  +--rw lkrInterval?   mplstpoamLkrInterval
         |  +--rw lkrEnable?     Enable
         .
         .
```

Figure 8 Snippet of data hierarchy related to ME of Fault Management

3.4.2. Performance Monitoring Configuration

The performance monitoring configuration is consist of seperate containers of delay, packet loss and jitter measurements.

```
|  +--rw meg* [megName]
         .
         .
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         .
         .
         +--rw te* [tunnelName tunnelId ingressLsrId]
         .
         .
         +--rw section* [sectionId]
         .
         .
         +--rw vllMip* [peerIp vcId switchPeerIp switchVcId vcType instanceName]
         .
         .
         +--rw teMip* [lspName]
         .
         .
         +--rw cc
         .
         .
         +--rw cv
         .
         .
         +--rw ais
         .
         .
         +--rw lkr
         .
         .
         +--rw oneWayDmSend
         |  +--rw oneDmSendEnable?   Enable
         |  +--rw oneDmInterval?     mplstpoamDmInterval
         |  +--rw oneDmExp?          uint8
         |  +--rw oneDmPacketSize?   uint16
         |  +--rw oneDmPadValue?     mplstpoamDmPaddingValue
         +--rw oneWayDmRcv
         |  +--rw oneDmRcvEnable?       Enable
         |  +--rw oneDmRcvEnableType?   mplstpoamOneWayRcvType
         +--rw twoWayDmSend
         |  +--rw twoDmSendEnable?   Enable
         |  +--rw twoDmInterval?     mplstpoamDmInterval
         |  +--rw twoDmExp?          uint8
         |  +--rw twoDmPacketSize?   uint16
         |  +--rw twoDmPadValue?     mplstpoamDmPaddingValue
         |  +--rw twoDmTimestamp?    Enable
         +--rw twoWayDmRcv
         |  +--rw twoDmRcvEnable?   Enable
         +--rw singleLmSend
         |  +--rw slmSendEnable?   Enable
         |  +--rw slmInterval?     mplstpoamSlmInterval
         |  +--rw slmExp?          uint8
         +--rw singleLmRcv
         |  +--rw slmRcvEnable?   Enable
         +--rw dualLm
         |  +--rw dlmEnable?   Enable
```

Figure 8 Snippet of data hierarchy related to MEG performance monitoring configuration

4. MPLS-TP OAM Data Hierarchy

The complete data hierarchy related to the MPLS-TP OAM YANG model is presented below. The following notations are used within the data tree and carry the meaning as below.

Each node is printed as:

<status> <flags> <name> <opts> <type>

<status> is one of:

+ for current

x for deprecated

o for obsolete

<flags> is one of:

rw for configuration data

ro for non-configuration data

-x for rpcs

-n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>.

<opts> is one of:

? for an optional leaf or choice

! for a presence container

* for a leaf-list or list

[<keys>] for a list’s keys

<type> is the name of the type for leafs and leaf-lists

```
module: mplstpoam
   +--rw global
   |  +--rw aisEnable?   Enable
   +--rw megs
      +--rw meg* [megName]
         +--rw megName             string
         +--rw meType?             mplstpoamMeType
         +--rw megId?              string
         +--rw megLevel?           uint8
         +--rw oamActiveState?     mplstpoamActiveType
         +--rw pw* [peerIp vcId vcType remotePeerIp remoteVcId remoteVcType]
         |  +--rw peerIp                    inet:ip-address
         |  +--rw vcId                      uint32
         |  +--rw vcType                    mplstpoamVctype
         |  +--rw remotePeerIp              inet:ip-address
         |  +--rw remoteVcId                mplstpoamVctype
         |  +--rw remoteVcType              mplstpoamVctype
         |  +--rw mepId?                    uint16
         |  +--rw remoteMepId?              uint16
         |  +--rw vllttl?                   uint8
         |  +--ro meIndex?                  uint32
         |  +--ro meDirection?              mplstpoamMeDirection
         |  +--ro meState?                  mplstpoamMeState
         |  +--ro localState?               mplstpoamMeState
         |  +--ro remoteState?              mplstpoamMeState
         |  +--ro alarmIndicate?            string
         |  +--ro localDefectStatus?        mplstpoamDefectStatusType
         |  +--ro localInvalidTime?         uint32
         |  +--ro localDefectLocation?      string
         |  +--ro localDefectType?          mplstpoamDefectType
         |  +--ro remoteDefectStatus?       mplstpoamDefectStatusType
         |  +--ro remoteInvalidTime?        uint32
         |  +--ro remoteDefectLocation?     string
         |  +--ro remoteDefectType?         mplstpoamDefectType
         |  +--rw galEnable?                Enable
         |  +--rw galMode?                  mplstpoamGalMode
         +--rw te* [tunnelName tunnelId ingressLsrId]
         |  +--rw tunnelName              string
         |  +--rw tunnelId                uint32
         |  +--rw ingressLsrId            inet:ip-address
         |  +--rw mepId?                  uint16
         |  +--rw remoteMepId?            uint16
         |  +--rw reverseTunnelName       string
         |  +--rw reverseTunnelId?        uint16
         |  +--rw reverseIngrLsrId?       inet:ip-address
         |  +--ro tunnelDescription?      string
         |  +--ro tunnelType?             mplstpoamTunnelType
         |  +--ro tunnelDirection?        mplstpoamTunnelDirectionType
         |  +--ro meIndex?                uint32
         |  +--ro meDirection?            mplstpoamMeDirection
         |  +--ro meState?                mplstpoamMeState
         |  +--ro localState?             mplstpoamMeState
         |  +--ro remoteState?            mplstpoamMeState
         |  +--ro alarmIndicate?          string
         |  +--ro localDefectStatus?      mplstpoamDefectStatusType
         |  +--ro localInvalidTime?       uint32
         |  +--ro localDefectLocation?    string
         |  +--ro localDefectType?        mplstpoamDefectType
         |  +--ro remoteDefectStatus?     mplstpoamDefectStatusType
         |  +--ro remoteInvalidTime?      uint32
         |  +--ro remoteDefectLocation?   string
         |  +--ro remoteDefectType?       mplstpoamDefectType
         |  +--ro meIndexEgress?          uint32
         |  +--ro meDirectEgress?         mplstpoamMeDirection
         |  +--ro statusBoardEgress?      string
         |  +--ro stateEgress?            mplstpoamMeState
         |  +--ro alarmEgress?            string
         +--rw section* [sectionId]
         |  +--rw sectionId               uint64
         |  +--rw ifName?                 string
         |  +--rw peerIp                  inet:ip-address
         |  +--rw peerLsrId?              inet:ip-address
         |  +--rw mepId?                  uint16
         |  +--rw remoteMepId?            uint16
         |  +--ro meIndex?                uint32
         |  +--ro meDirection?            mplstpoamMeDirection
         |  +--ro meState?                mplstpoamMeState
         |  +--ro localState?             mplstpoamMeState
         |  +--ro remoteState?            mplstpoamMeState
         |  +--ro alarmIndicate?          string
         |  +--ro localDefectStatus?      mplstpoamDefectStatusType
         |  +--ro localInvalidTime?       uint32
         |  +--ro localDefectLocation?    string
         |  +--ro localDefectType?        mplstpoamDefectType
         |  +--ro remoteDefectStatus?     mplstpoamDefectStatusType
         |  +--ro remoteInvalidTime?      uint32
         |  +--ro remoteDefectLocation?   string
         |  +--ro remoteDefectType?       mplstpoamDefectType
         +--rw vllMip* [peerIp vcId switchPeerIp switchVcId vcType instanceName]

         |  +--rw peerIp          inet:ip-address
         |  +--rw vcId            uint32
         |  +--rw switchPeerIp    inet:ip-address
         |  +--rw switchVcId      uint32
         |  +--rw vcType          mplstpoamVctype
         |  +--rw instanceName    string
         |  +--ro meIndex?        uint32
         |  +--ro meDirection?    mplstpoamMeDirection
         |  +--ro meState?        mplstpoamMeState
         |  +--rw mipId?          uint16
         +--rw teMip* [lspName]
         |  +--rw lspName        string
         |  +--ro meIndex?       uint32
         |  +--ro meDirection?   mplstpoamMeDirection
         |  +--ro meState?       mplstpoamMeState
         |  +--rw mipId?         uint16
         +--rw cc
         |  +--rw ccSessionMode?            mplstpoamCcSessionMode
         |  +--rw ccAuthenticationEnable?   Enable
         |  +--rw ccExp?                    uint8
         |  +--rw ccTransmitInterval?       mplstpoamCcInterval
         |  +--rw ccRecieveInterval?        mplstpoamCcInterval
         |  +--rw ccDetectMultiplier?       mplstpoamCcDetectMultiplier
         |  +--rw ccEnable?                 Enable
         +--rw cv
         |  +--rw cvSessionMode?            mplstpoamCcSessionMode
         |  +--rw cvAuthenticationEnable?   Enable
         |  +--rw cvExp?                    uint8
         |  +--rw cvInterval?               mplstpoamCvInterval
         |  +--rw cvDetectMultiplier?       mplstpoamCvDetectMultiplier
         |  +--rw cvEnable?                 Enable
         +--rw ais
         |  +--rw aisExp?        uint8
         |  +--rw aisInterval?   mplstpoamAisInterval
         +--rw lkr
         |  +--rw lkrExp?        uint8
         |  +--rw lkrInterval?   mplstpoamLkrInterval
         |  +--rw lkrEnable?     Enable
         +--rw oneWayDmSend
         |  +--rw oneDmSendEnable?   Enable
         |  +--rw oneDmInterval?     mplstpoamDmInterval
         |  +--rw oneDmExp?          uint8
         |  +--rw oneDmPacketSize?   uint16
         |  +--rw oneDmPadValue?     mplstpoamDmPaddingValue
         +--rw oneWayDmRcv
         |  +--rw oneDmRcvEnable?       Enable
         |  +--rw oneDmRcvEnableType?   mplstpoamOneWayRcvType
         +--rw twoWayDmSend
         |  +--rw twoDmSendEnable?   Enable
         |  +--rw twoDmInterval?     mplstpoamDmInterval
         |  +--rw twoDmExp?          uint8
         |  +--rw twoDmPacketSize?   uint16
         |  +--rw twoDmPadValue?     mplstpoamDmPaddingValue
         |  +--rw twoDmTimestamp?    Enable
         +--rw twoWayDmRcv
         |  +--rw twoDmRcvEnable?   Enable
         +--rw singleLmSend
         |  +--rw slmSendEnable?   Enable
         |  +--rw slmInterval?     mplstpoamSlmInterval
         |  +--rw slmExp?          uint8
         +--rw singleLmRcv
         |  +--rw slmRcvEnable?   Enable
         +--rw dualLm
         |  +--rw dlmEnable?   Enable
         +--ro oneWayDmResult
         |  +--ro sendPktNum?      uint32
         |  +--ro recvPktNum?      uint32
         |  +--ro delayMin?        uint32
         |  +--ro delayMax?        uint32
         |  +--ro delayAvg?        uint32
         |  +--ro jitterMin?       uint32
         |  +--ro jitterMax?       uint32
         |  +--ro jitterAvg?       uint32
         |  +--ro oneWayDmDatas
         |     +--ro oneWayDmData* [index]
         |        +--ro index          uint32
         |        +--ro oneDelay?      uint32
         |        +--ro oneDelayVar?   uint32
         |        +--ro errorInfo?     mplstpoamErrorInfo
         +--ro oneWaySendResult
         |  +--ro measureMode?   mplstpoamMeasureMode
         |  +--ro status?        mplstpoamStatisticsStatus
         +--ro twoWayDmResult
         |  +--ro measureMode?     mplstpoamMeasureMode
         |  +--ro status?          mplstpoamStatisticsStatus
         |  +--ro sendPktNum?      uint32
         |  +--ro recvPktNum?      uint32
         |  +--ro delayMin?        uint32
         |  +--ro delayMax?        uint32
         |  +--ro delayAvg?        uint32
         |  +--ro jitterMin?       uint32
         |  +--ro jitterMax?       uint32
         |  +--ro jitterAvg?       uint32
         |  +--ro twoWayDmDatas
         |     +--ro twoWayDmData* [index]
         |        +--ro index          uint32
         |        +--ro twoDelay?      uint32
         |        +--ro twoDelayVar?   uint32
         |        +--ro errorInfo?     mplstpoamErrorInfo
         +--ro singleLmResult
         |  +--ro measureMode?       mplstpoamMeasureMode
         |  +--ro status?            mplstpoamStatisticsStatus
         |  +--ro sendPktNum?        uint32
         |  +--ro recvPktNum?        uint32
         |  +--ro rmtLossRatioMin?   uint32
         |  +--ro rmtLossRatioMax?   uint32
         |  +--ro rmtLossRatioAvg?   uint32
         |  +--ro rmtLossCountMin?   uint32
         |  +--ro rmtLossCountMax?   uint32
         |  +--ro rmtLossCountAvg?   uint32
         |  +--ro singleLmDatas
         |     +--ro singleLmData* [index]
         |        +--ro index            uint32
         |        +--ro slmLossLcl?      uint32
         |        +--ro slmLossLclRat?   string
         |        +--ro slmLossRmt?      uint32
         |        +--ro slmLossRmtRat?   string
         |        +--ro errorInfo?       mplstpoamErrorInfo
         +--ro dualLmDatas
            +--ro dualLmData* [index]
               +--ro index            uint32
               +--ro dlmLossLcl?      uint32
               +--ro dlmLossLclRat?   string
               +--ro dlmLossRmt?      uint32
               +--ro dlmLossRmtRat?   string
               +--ro errorInfo?       mplstpoamErrorInfo
```

5. MPLS-TP OAM YANG module

```
module mplstpoam {
    namespace "urn:ietf:params:xml:ns:yang:mplstpoam";
    //namespace need to be assigned by IANA
    prefix "mplstpoam";
    import ietf-inet-types {
    prefix inet;
    }
    organization "IETF MPLS (Multiprotocol Label Switching) Working Group";
    contact "monica.zhangli@huawei.com
```

```
        vero.zheng@huawei.com
        aldrin.ietf@gmail.com";
    description "MPLS TP OAM Yang Module";
    revision "2014-09-18";

    typedef Enable {
        type enumeration {
            enum "true" {
                value 0;
            }
            enum "false" {
                value 1;
            }
        }
    }
    typedef mplstpoamMeType {
        type enumeration {
            enum "none" {
                value 0;
                description "ME type is valid";
            }
            enum "vll" {
                value 1;
                description "ME type is vll";
            }
            enum "vpls" {
                value 2;
                description "ME type is vpls";
            }
            enum "ingress" {
                value 3;
                description "ME type is ingress";
            }
            enum "egress" {
                value 4;
                description "ME type is egress";
            }
            enum "co-route" {
                value 5;
                description "ME type is co-route bidirectional te";
            }
            enum "associate" {
                value 6;
                description "ME type is associate bidirectional te";
            }
            enum "section" {
                value 7;
                description "ME type is section";
            }
            enum "vllMip" {
            value 8;
                description "ME type is vllMip";
            }
            enum "teMip" {
                value 9;
                description "ME type is teMip";
            }
        }
    }
    typedef mplstpoamTeServiceType {
        type enumeration {
            enum "none" {
                value 0;
                description "Service type is valid";
            }
            enum "te-crlsp" {
                value 1;
                description "Service type is te-crlsp";
            }
            enum "associate-te-lsp" {
                value 2;
                description "Service type is associate-te-lsp";
            }
            enum "te-ingress" {
                value 3;
                description "Service type is te-ingress";
            }
            enum "te-egress" {
                value 4;
                description "Service type is te-egresst";
            }
        }
    }
    typedef mplstpoamCcSessionMode {
        type enumeration {
            enum "coordinated" {
                value 0;
            }
            enum "independent" {
                value 1;
            }
        }
    }
    typedef mplstpoamCcInterval {
        description "The value rang for cc packet transmit and receive interval"
;
        type uint32{
```

```
                       range "1..65535";
        }
    }
    typedef mplstpoamCvInterval {
        description "The value rang for cv packet transmit interval";
        type uint32{
                       range "1..65535";
        }
    }
    typedef mplstpoamCcDetectMultiplier {
        description "The value rang for cv packet detect multiplier";
        type uint8{
                       range "1..255";
        }
    }
    typedef mplstpoamCvDetectMultiplier {
        description "The value rang for cv packet detect multiplier";
        type uint8{
                       range "1..255";
        }
    }
    typedef mplstpoamLkrInterval {
        type enumeration {
            enum "interval1000ms" {
                value 0;
            }
            enum "interval60000ms" {
                value 1;
            }
        }
    }
    typedef mplstpoamAisInterval {
        type enumeration {
            enum "interval1000ms" {
                value 0;
            }
            enum "interval60000ms" {
                value 1;
            }
        }
    }
    typedef mplstpoamMeDirection {
        type enumeration {
            enum "ingress" {
                value 0;
                description "The direction to the ME is ingress";
            }
            enum "egress" {
                value 1;
                description "The direction to the ME is egress";
            }
            enum "dual" {
                value 2;
                description "The direction to the ME is dual";
            }
            enum "none" {
                value 3;
                description "The direction to the ME is none";
            }
        }
    }
    typedef mplstpoamMeState {
        type enumeration {
            enum "init" {
                value 0;
                    description "The me state is init";
            }
            enum "down" {
                value 1;
                description "The me state is down";
            }
            enum "up" {
                value 2;
                description "The me state is up";
            }
        }
    }
    typedef mplstpoamDmInterval {
        description "The value rang for dm packet transmit interval";
        type uint32 {
                       range "1..65535";
        }
    }
    typedef mplstpoamDmPaddingValue {
        type enumeration {
            enum "paddingvalue0" {
                value 0;
            }
            enum "paddingvalue1" {
                value 1;
            }
        }
    }
    typedef mplstpoamSlmInterval {
        description "The value rang for lm packet transmit interval";
        type uint32 {
```

```
                       range "1..65535";
        }
    }
    typedef mplstpoamMeasureMode {
        type enumeration {
            enum "on-demand" {
                value 0;
            }
            enum "proactive" {
                value 1;
            }
        }
    }
    typedef mplstpoamVctype {
        description "The namespace of the vc type of pw";
        type string {
                       length "1..8191";
        }
    }
    typedef mplstpoamStatisticsStatus {
        type enumeration {
            enum "finished" {
                value 0;
            }
            enum "working" {
                value 1;
            }
        }
    }
    typedef mplstpoamErrorInfo {
        type enumeration {
            enum "valid" {
                value 0;
            }
            enum "invalid-loss" {
                value 1;
            }
            enum "invalid-delay" {
                value 2;
            }
        }
    }
    typedef mplstpoamDefectStatusType {
            description "The namespace of defect status type";
        type string {
                       length "1..8191";
        }
    }
    typedef mplstpoamDefectType {
            description "The namespace of defect type";
        type string {
                       length "1..8191";
        }
    }
    typedef mplstpoamTunnelType {
        type enumeration {
            enum "ingress" {
                value 0;
            }
            enum "egress" {
                value 1;
            }
            enum "bidirectional" {
                value 2;
            }
        }
    }
    typedef mplstpoamTunnelDirectionType {
        type enumeration {
            enum "uniDrectional" {
                value 0;
            }
            enum "biDirectional" {
                value 1;
            }
        }
    }
    typedef mplstpoamActiveType {
        type enumeration {
            enum "deactive" {
                value 0;
            }
            enum "active" {
                value 1;
            }
        }
    }
    typedef mplstpoamGalMode {
        type enumeration {
            enum "with-13" {
                value 0;
                description "Gal mode is with label 13";
            }
            enum "without-13" {
                value 1;
                description "Gal mode is without label 13";
            }
        }
    }
    typedef mplstpoamOneWayRcvType {
        type enumeration {
            enum "on-demand" {
                value 0;
                description "The switch of receive eanble takes effect on-demand
 one-way delay-measure";
            }
            enum "proactive" {
                value 1;
                description "The switch of receive eanble takes effect proactive
 one-way delay-measure";
            }
        }
    }
    grouping ME-detect-status {
        description "This node indicate detect status of ME";
        leaf meIndex {
            description "The object indicates the index of ME";
            config "false";
            type uint32 {
                range "1..65535";
            }
        }
        leaf meDirection {
            description "The object indicates the direction of ME";
            config "false";
            type mplstpoamMeDirection;
        }
        leaf meState {
            description "The object indicates the state of ME";
            config "false";
            type mplstpoamMeState;
        }
        leaf localState {
            description "The object indicates the local status of ME";
            config "false";
            type mplstpoamMeState;
        }
        leaf remoteState {
            description "The object indicates the remote state of ME";
            config "false";
            type mplstpoamMeState;
        }
        leaf alarmIndicate {
            description "The object indicates the alarm of ME";
            config "false";
            type string {
                length "1..26";
            }
        }
        leaf localDefectStatus {
            description "This object indicates the local defect status";
            config "false";
            default "init";
            type mplstpoamDefectStatusType;
        }
        leaf localInvalidTime {
            description "This object indicates the invalid Time of local detect"
;
            config "false";
            type uint32 {
                range "0..4294967295";
            }
        }
        leaf localDefectLocation {
            description "This object indicates the local defect location";
            config "false";
            type string {
                length "1..32";
            }
        }
        leaf localDefectType {
            description "This object indicates the local defect type";
            config "false";
            type mplstpoamDefectType;
        }
        leaf remoteDefectStatus {
            description "This object indicates the remote defect status";
            config "false";
            default "init";
            type mplstpoamDefectStatusType;
        }
        leaf remoteInvalidTime {
            description "This object indicates the invalid Time of remote detect
";
            config "false";
            type uint32 {
                range "0..4294967295";
            }
        }
        leaf remoteDefectLocation {
            description "This object indicates the remote defect location";
            config "false";
            type string {
                length "1..32";
            }
        }
        leaf remoteDefectType {
            description "This object indicates the remote defect type";
            config "false";
            type mplstpoamDefectType;
        }

    }
    grouping gal-set {
        description "This object indicates the gal set";
        leaf galEnable {
            description "This object indicates the gal flag";
            config "true";
            default "true";
            type Enable;
        }
        leaf galMode {
            description "This object indicates the gal flag";
            config "true";
            type mplstpoamGalMode;
        }
    }

    container global {

        leaf aisEnable {
            description "This object indicates the global ais flag of mpls-tp oa
m";
            config "true";
            default "false";
            type Enable;
        }
    }
    container megs {
        status current;
        description "show the megs";

        list meg {

            key "megName";
            leaf megName {
                description "The object indicates the name of MEG";
                config "true";
                mandatory "true";
                type string {
                    length "1..14";
                }
            }
            leaf meType {
                description "The object indicates the type of ME";
                config "true";
                default "none";
                type mplstpoamMeType;
            }
            leaf megId {
                description "The object indicates the ID of MEG";
                config "true";
                type string {
                    length "1..96";
                }
            }
            leaf megLevel {
                description "The object indicates the level of MEG";
                config "true";
                default "7";
                type uint8 {
                    range "0..7";
                }
            }
            leaf oamActiveState {
                description "This object indicates the oam active state";
                config "true";
                default "deactive";
                type mplstpoamActiveType;
            }
            list pw {

                key "peerIp vcId vcType remotePeerIp remoteVcId remoteVcType";
                leaf peerIp {
                    description "This object indicates the peer IP address";
                    config "true";
                    mandatory "true";
                    type inet:ip-address;
                }
                leaf vcId {
                    description "This object indicates the vc ID of PW type ME";
                    config "true";
                    mandatory "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf vcType {
                    description "This object indicates the vc type of VC type ME
";
                    config "true";
                    mandatory "true";
                    type mplstpoamVctype;
                }
                leaf remotePeerIp {
                    description "This object indicates the remote peer IP of PW
type ME";
                    config "true";
                    type inet:ip-address;
                }
                leaf remoteVcId {
                    description "This object indicates the remote vc ID of PW ty
pe ME";
                    config "true";
                    type mplstpoamVctype;
                }
                leaf remoteVcType {
                    description "This object indicates the remote vc type of PW
type ME";
                    config "true";
                    type mplstpoamVctype;
                }
                leaf mepId {
                    description "This object indicates the MEP Id of local ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                leaf remoteMepId {
                    description "This object indicates the MEP Id of remote ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                leaf vllttl {
                    description "This object indicates the VLL ttl of PW type ME
";
                    config "true";
                    type uint8 {
                        range "1..255";
                    }
                }
                uses ME-detect-status;
                uses gal-set;
            }

            list te {
                key "tunnelName tunnelId ingressLsrId";
                leaf tunnelName  {
                    description "The object indicates the name of tunnel";
                    config "true";
                    mandatory "true";
                    type string {
                        length "0..63";
                    }
                }
                leaf tunnelId {
                    description "The object indicates the tunnel id";
                    config "true";
                    type uint32 {
                        range "1..65535";
                    }
                }
                leaf ingressLsrId {
                    description "The object indicates the ingress LSR-ID";
                    config "true";
                    type inet:ip-address;
                }
                leaf mepId {
                    description "This object indicates the MEP Id of local ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                leaf remoteMepId {
                    description "This object indicates the MEP Id of remote ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                leaf reverseTunnelName  {
                    description "The object indicates the name of reverse tunnel
";

                    config "true";
                    mandatory "true";
                    type string {
                        length "0..63";
                    }
                }

                leaf reverseTunnelId {
                    description "The object indicates the ingress reverse tunnel
Id";

                    config "true";
                    default "10";
                    type uint16 {
                        range "1..65535";
                    }
                }
                leaf reverseIngrLsrId {
                    description "The object indicates the ingress reverse LSR-ID
";

                    config "true";
                    type inet:ip-address;
                }
                leaf tunnelDescription {
                    description "The object indicates the description of tunnel"
;
                    config "false";
                    type string {
                        length "1..32";
                    }
                }
                leaf tunnelType {
                    description "The object indicates the type of tunnel";
                    config "false";
                    default "ingress";
                    type mplstpoamTunnelType;
                }
                leaf tunnelDirection {
                    description "The object indicates the direction of tunnel";
                    config "false";
                    type mplstpoamTunnelDirectionType;
                }
                uses ME-detect-status;
                leaf meIndexEgress {
                    description "The object indicates the egress index of ME";
                    config "false";
                    type uint32 {
                        range "1..65535";
                    }
                }
                leaf meDirectEgress {
                    description "The object indicates the direction of egress ME
";
                    config "false";
                    type mplstpoamMeDirection;
                }
                leaf statusBoardEgress {
                    description "The object indicates the selected status board
of ME";
                    config "false";
                    type string {
                        length "1..19";
                    }
                }
                leaf stateEgress {
                    description "The object indicates the status of ME";
                    config "false";
                    type mplstpoamMeState;
                }
                leaf alarmEgress {
                    description "The object indicates the alarm of ME";
                    config "false";
                    type string {
                        length "1..26";
                    }
                }
            }

            list section {

                key "sectionId";
                leaf sectionId {
                    description "This object indicates the section ID";
                    config "true";
                    type uint64 {
                        range "1..2147483647";
                    }
                }
                leaf ifName {
                    description "The object indicates the interface name";
                    config "true";
                    type string {
                        length "1..63";
                    }
                }
                leaf peerIp {
                    description "This object indicates the peer IP address";
                    config "true";
                    mandatory "true";
                    type inet:ip-address;
                }
                leaf peerLsrId {
                    description "This object indicates the peer lsr ID";
                    config "true";
                    type inet:ip-address;
                }
                leaf mepId {
                    description "This object indicates the MEP Id of local ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                leaf remoteMepId {
                    description "This object indicates the MEP Id of remote ME";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
                uses ME-detect-status;
            }

            list vllMip {

                key "peerIp vcId switchPeerIp switchVcId vcType instanceName";
                leaf peerIp {
                    description "This object indicates the peer IP address of PW
 type MIP";
                    config "true";
                    mandatory "true";
                    type inet:ip-address;
                }
                leaf vcId {
                    description "This object indicates the vc ID of PW type MIP"
;
                    config "true";
                    mandatory "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf switchPeerIp {
                    description "This object indicates the peer IP address of PW
 switch node";
                    config "true";
                    mandatory "true";
                    type inet:ip-address;
                }
                leaf switchVcId {
                    description "This object indicates the vc id of PW switch no
de";
                    config "true";
                    mandatory "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf vcType {
                    description "This object indicates the vc type of PW type MI
P";
                    config "true";
                    mandatory "true";
                    type mplstpoamVctype;
                }
                leaf instanceName {
                    description "This object specifies the VPWS instance name";
                    config "true";
                    mandatory "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf meIndex {
                    description "The object indicates the index of MIP";
                    config "false";
                    type uint32 {
                        range "1..65535";
                    }
                }
                leaf meDirection {
                    description "The object indicates the direction of MIP";
                    config "false";
                    type mplstpoamMeDirection;
                }
                leaf meState {
                    description "The object indicates the state of MIP";
                    config "false";
                    type mplstpoamMeState;
                }
                leaf mipId {
                    description "The object indicates the ID of MIP";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
            }

            list teMip {

                key "lspName";

                leaf lspName {
                    description "This object indicates the name of LSP";
                    type string {
                        length "1..16";
                    }
                }

                leaf meIndex {
                    description "The object indicates the index of te MIP";
                    config "false";
                    type uint32 {
                        range "1..65535";
                    }
                }
                leaf meDirection {
                    description "The object indicates the direction of te MIP";
                    config "false";
                    type mplstpoamMeDirection;
                }
                leaf meState {
                    description "The object indicates the state of te MIP";
                    config "false";
                    type mplstpoamMeState;
                }
                leaf mipId {
                    description "The object indicates the ID of te MIP";
                    config "true";
                    type uint16 {
                        range "1..8191";
                    }
                }
            }

            container cc {

                leaf ccSessionMode {
                     description "This object indicates the session mode of CC";
                     config "true";
                     default "coordinated";
                         type mplstpoamCcSessionMode;
                }
                leaf ccAuthenticationEnable {
                    config "true";
                    default "true";
                    type Enable;
                }
                leaf ccExp {
                    description "This object indicates the exp of CC packet whic
h is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf ccTransmitInterval {
                    description "The interval of CC packet which is transmit in
the MEG";
                    config "true";
                    default "1";
                    type mplstpoamCcInterval;
                }
                leaf ccRecieveInterval {
                    description "The interval of CC packet which is recieved in
the MEG";
                    config "true";
                    default "1";
                    type mplstpoamCcInterval;
                }
                leaf ccDetectMultiplier {
                    description "The object indicate the detect multiplier of CC
 packet";
                    config "true";
                    default "3";
                    type mplstpoamCcDetectMultiplier;
                }
                leaf ccEnable {
                    description "The object indicates whether CC can be sent by
the MEG";
                    config "true";
                    default "true";
                    type Enable;
                }
            }
            container cv {
                    leaf cvSessionMode {
                     description "This object indicates the session mode of CC";
                     config "true";
                     default "coordinated";
                         type mplstpoamCcSessionMode;
                }
                leaf cvAuthenticationEnable {
                    config "true";
                    default "true";
                    type Enable;
                }
                leaf cvExp {
                    description "This object indicates the exp of CV packet whic
h is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                    leaf cvInterval {
                    description "The interval of CV packet which is sent in the
MEG";
                    config "true";
                    default "1";
                    type mplstpoamCvInterval;
                }
                leaf cvDetectMultiplier {
                    description "The object indicate the detect multiplier of CV
 packet";
                    config "true";
                    default "3";
                    type mplstpoamCvDetectMultiplier;
                }
                leaf cvEnable {
                    description "The object indicates whether CC can be received
 by the MEG";
                    config "true";
                    default "true";
                    type Enable;
                }

            }

            container ais {

                config "true";

                leaf aisExp {
                    description "This object indicates the exp of AIS packet whi
ch is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf aisInterval {
                    description "This object indicates the interval of AIS packe
t which is sent in the MEG";
                    config "true";
                    default "interval1000ms";
                    type mplstpoamAisInterval;
                }
            }

            container lkr {

                config "true";

                leaf lkrExp {
                    description "This object indicates the exp of lock report pa
cket which is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf lkrInterval {
                    description "This object indicates the interval of lock repo
rt packet which is sent in the MEG";
                    config "true";
                    default "interval1000ms";
                    type mplstpoamLkrInterval;
                }
                leaf lkrEnable {
                    description "The object indicates whether lock report is ena
bled in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
            }
            container oneWayDmSend {


                leaf oneDmSendEnable {
                    description "This object indicates the 1DM statistics is ena
bled in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
                leaf oneDmInterval {
                    description "This object indicates the interval of 1DM stati
stics in the MEG";
                    config "true";
                    default "1000";
                    type mplstpoamDmInterval;
                }
                leaf oneDmExp {
                    description "This object indicates the exp of 1DM packet whi
ch is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf oneDmPacketSize {
                    description "This object indicates the packet size of 1DM pa
cket which is sent in the MEG";
                    config "true";
                    type uint16 {
                        range "64..1518";
                    }
                }
                leaf oneDmPadValue {
                    description "This object indicates the padding value of 1DM
packet which is sent in the MEG";
                    config "true";
                    default "paddingvalue0";
                    type mplstpoamDmPaddingValue;
                }
            }

            container oneWayDmRcv {


                leaf oneDmRcvEnable {
                    description "This object indicates the 1DM receive is enable
d in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
                leaf oneDmRcvEnableType {
                    description "This object indicates the 1DM receive type";
                    config "true";
                    type mplstpoamOneWayRcvType;
                }
            }

            container twoWayDmSend {

                leaf twoDmSendEnable {
                    description "This object indicates the 2DM statistics is ena
bled in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
                leaf twoDmInterval {
                    description "This object indicates the interval of 2DM stati
stics in the MEG";
                    config "true";
                    default "1000";
                    type mplstpoamDmInterval;
                }
                leaf twoDmExp {
                    description "This object indicates the exp of 2DM packet whi
ch is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
                leaf twoDmPacketSize {
                    description "This object indicates the packet size of 2DM pa
cket which is sent in the MEG";
                    config "true";
                    type uint16 {
                        range "64..1518";
                    }
                }
                leaf twoDmPadValue {
                    description "This object indicates the padding value of 2DM
packet which is sent in the MEG";
                    config "true";
                    default "paddingvalue0";
                    type mplstpoamDmPaddingValue;
                }
                leaf twoDmTimestamp {
                    description "This object indicates whether two-way delay mea
surement time stamp is enable in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
            }

            container twoWayDmRcv {

                leaf twoDmRcvEnable {
                    description "This object indicates the 2DM receiving statist
ics is enabled in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
            }

            container singleLmSend {

                leaf slmSendEnable {
                    description "This object indicates whether slm send is enabl
e in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
                leaf slmInterval {
                    description "This object indicates the interval of slm stati
stics in the MEG";
                    config "true";
                    default "1000";
                    type mplstpoamSlmInterval;
                }
                leaf slmExp {
                    description "This object indicates the exp of slm packet whi
ch is sent in the MEG";
                    config "true";
                    default "7";
                    type uint8 {
                        range "0..7";
                    }
                }
            }

            container singleLmRcv {

                leaf slmRcvEnable {
                    description "This object indicates whether slm receive is en
able in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
            }
            container dualLm {

                leaf dlmEnable {
                    description "This object indicates the dual loss statistics
is enabled in the MEG";
                    config "true";
                    default "false";
                    type Enable;
                }
            }

            container oneWayDmResult {
                config "false";

                leaf sendPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf recvPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayMin {
                    description "This object indicates the minimum delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayMax {
                    description "This object indicates the maximum delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayAvg {
                    description "This object indicates the average delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterMin {
                    description "This object indicates the minimum jitter of rec
eived LB packets in the MEG";
```

```
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterMax {
                    description "This object indicates the average jitter of rec
eived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterAvg {
                    description "This object indicates the average jitter of rec
eived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                container oneWayDmDatas {
                    config "false";

                    list oneWayDmData {

                        key "index";
                        config "false";

                        leaf index {
                            description "This object indicates index of 1DM stat
istics record in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf oneDelay {
                            description "This object indicates delay of 1DM stat
istics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf oneDelayVar {
                            description "This object indicates delay Variation o
f 1DM statistics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf errorInfo {
                            description "This object indicates the error info of
 statistics record in the MEG";
                            config "false";
                            type mplstpoamErrorInfo;
                        }
                    }

                }

            }

            container oneWaySendResult {
                config "false";

                leaf measureMode {
                    description "The flag indicates whether the measurement is a
n on-demand or a continue measurement";
                    config "false";
                    default "on-demand";
                    type mplstpoamMeasureMode;
                }
                leaf status {
                    description "The flag indicates whether the measurement is f
inished";
                    config "false";
                    default "finished";
                    type mplstpoamStatisticsStatus;
                }
            }

            container twoWayDmResult {
                config "false";

                leaf measureMode {
                    description "The flag indicates whether the measurement is a
n on-demand or a continue measurement";
                    config "false";
                    default "on-demand";
                    type mplstpoamMeasureMode;
                }
                leaf status {
                    description "The flag indicates whether the measurement is f
inished";
                    config "false";
                    default "finished";
                    type mplstpoamStatisticsStatus;
                }
                leaf sendPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf recvPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayMin {
                    description "This object indicates the minimum delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayMax {
                    description "This object indicates the maximum delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf delayAvg {
                    description "This object indicates the average delay of rece
ived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterMin {
                    description "This object indicates the minimum jitter of rec
eived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterMax {
                    description "This object indicates the average jitter of rec
eived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf jitterAvg {
                    description "This object indicates the average jitter of rec
eived LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                container twoWayDmDatas {
                    config "false";

                    list twoWayDmData {

                        key "index";
                        config "false";

                        leaf index {
                            description "This object indicates index of 2DM stat
istics record in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf twoDelay {
                            description "This object indicates delay of 2DM stat
istics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf twoDelayVar {
                            description "This object indicates delay Variation o
f 2DM statistics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf errorInfo {
                            description "This object indicates the error info of
 statistics record in the MEG";
                            config "false";
                            type mplstpoamErrorInfo;
                        }
                    }

                }

            }

            container singleLmResult {

                config "false";

                leaf measureMode {
                    description "The flag indicates whether the measurement is a
n on-demand or a continue measurement";
                    config "false";
                    default "on-demand";
                    type mplstpoamMeasureMode;
                }
                leaf status {
                    description "The flag indicates whether the measurement is f
inished";
                    config "false";
                    default "finished";
                    type mplstpoamStatisticsStatus;
                }
                leaf sendPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf recvPktNum {
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossRatioMin {
                    description "This object indicates the minimum loss-ratio of
 received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossRatioMax {
                    description "This object indicates the maximum loss-ratio of
 received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossRatioAvg {
                    description "This object indicates the average loss-ratio of
 received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossCountMin {
                    description "This object indicates the minimum packet lost o
f received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossCountMax {
                    description "This object indicates the average packet lost o
f received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf rmtLossCountAvg {
                    description "This object indicates the average packet lost o
f received LB packets in the MEG";
                    config "false";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                container singleLmDatas {

                    config "false";

                    list singleLmData {

                        key "index";
                        config "false";

                        leaf index {
                            description "This object indicates index of slm stat
istics record in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf slmLossLcl {
                            description "This object indicates local packet loss
 of slm statistics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf slmLossLclRat {
                            description "This object indicates local packet loss
 rate of slm statistics in the MEG";
                            config "false";
                            type string {
                                length "1..24";
                            }
                        }
                        leaf slmLossRmt {
                            description "This object indicates remote packet los
s of slm statistics in the MEG";
                            config "false";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf slmLossRmtRat {
                            description "This object indicates remote packet los
s rate of slm statistics in the MEG";
                            config "false";
                            type string {
                                length "1..24";
                            }
                        }
                        leaf errorInfo {
                            description "This object indicates the error info of
 statistics record in the MEG";
                            config "false";
                            type mplstpoamErrorInfo;
                        }
                    }

                }

            }

            container dualLmDatas {

                config "false";

                list dualLmData {

                    key "index";
                    config "false";

                    leaf index {
                        description "This object indicates index of dlm statisti
cs record in the MEG";
                        config "false";
                        type uint32 {
                            range "1..4294967295";
                        }
                    }
                    leaf dlmLossLcl {
                        description "This object indicates local packet loss of
dlm statistics in the MEG";
                        config "false";
                        type uint32 {
                            range "1..4294967295";
                        }
                    }
                    leaf dlmLossLclRat {
                        description "This object indicates local packet loss rat
e of dlm statistics in the MEG";
                        config "false";
                        type string {
                            length "1..24";
                        }
                    }
                    leaf dlmLossRmt {
                        description "This object indicates remote packet loss of
 dlm statistics in the MEG";
                        config "false";
                        type uint32 {
                            range "1..4294967295";
                        }
                    }
                    leaf dlmLossRmtRat {
                        description "This object indicates remote packet loss ra
te of dlm statistics in the MEG";
                        config "false";
                        type string {
                            length "1..24";
                        }
                    }
                    leaf errorInfo {
                        description "This object indicates the error info of sta
tistics record in the MEG";
                        config "false";
                        type mplstpoamErrorInfo;
                    }
                }

            }

        }

    }
}
```

6. Security Considerations

TBD

7. IANA Considerations

This document registers the following namespace URI in the IETF XML registry.

URI:TBD

8. Acknowledgements

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, September 2011.

[RFC6375] Frost, D. and S. Bryant, "A Packet Loss and Delay Measurement Profile for MPLS-Based Transport Networks", RFC 6375, September 2011.

[RFC6427] Swallow, G., Fulignoli, A., Vigoureux, M., Boutros, S., and D. Ward, "MPLS Fault Management Operations, Administration, and Maintenance (OAM)", RFC 6427, November 2011.

[RFC6428] Allan, D., Swallow Ed. , G., and J. Drake Ed. , "Proactive Connectivity Verification, Continuity Check, and Remote Defect Indication for the MPLS Transport Profile", RFC 6428, November 2011.

9.2. Infomative References

[RFC6371] Busi, I. and D. Allan, "Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks", RFC 6371, September 2011.

Authors’ Addresses

Li Zhang
Huawei Technologies
China

Email: monica.zhangli@huawei.com

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Sam K. Aldrin
Huawei Technologies
USA

Email: aldrin.ietf@gmail.com