MPLS
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2015

S. Bryant
G. Swallow
S. Sivabalan
Cisco Systems

March 2, 2015

RFC6374 over UDP
draft-bryant-mpls-rfc6374-over-udp-00

Abstract

In draft-bryant-mpls-synonymous-flow-labels the concept of MPLS synonymous flow labels (SFL) was introduced and it was shown how they could be used to support RFC6374 loss measurements. In draft-bryant-mpls-sfl-control the request, lifetime management and withdrawal of SFLs was described. In this memo we show how these two protocols can be run over UDP to support the operation of RFC6374 in systems that do not support the Generic Associated Channel Label (GAL) (RFC5586).

Status of This Memo

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

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

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

This Internet-Draft will expire on September 3, 2015.

Copyright Notice

Table of Contents

1. Introduction 2
2. Requirements Language 3
3. Protocol Stack 3
3.1. Querier to Responder 3
4. Manageability Considerations 4
5. Security Considerations 4
6. IANA Considerations 4
7. Acknowledgements 5
8. Normative References 5
Authors' Addresses 5

1. Introduction

In draft-bryant-mpls-synonymous-flow-labels the concept of MPLS synonymous flow labels (SFL) was introduced and it was shown how they could be used to support RFC6374 loss measurements. In draft-bryant-mpls-sfl-control the request, lifetime management and withdrawal of SFLs was described. In this memo we show how these two protocols can be run over UDP to support the operation of RFC6374 in systems that do not support the Generic Associated Channel Label (GAL) [RFC5586].

The approach is to run an Associated Channel Header directly over UDP using the ACH UDP port supplemented by addressing information carried in the ACH payload. This memo explains how the extension of RFC6374 as described in draft-bryant-mpls-synonymous-flow-labels and draft-bryant-mpls-sfl-control provide the necessary information to provide mapping between the RFC6374 packet carried over UDP and the MPLS construct being monitored, even when the RFC6374 protocol exchange is entirely out of band relative to the Label Switched Path (LSP), Virtual Private Network (VPN) or Pseudowire (PW) being instrumented.

The key to this is the decoupling between the RFC6374 message and the data plane provided through the use of synonymous flow labels (SFL) as described in draft-bryant-mpls-synonymous-flow-labels.

Nothing in this memo prevents the use of the ACH UDP port for other types of Associated Channels, but the precise method of doing so is outside the scope of this text.

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].

3. Protocol Stack

The protocol stack is shown in Figure 1. It consists of three components, the UDP header, the ACH and either an RFC6374 message or an SFL Control message.

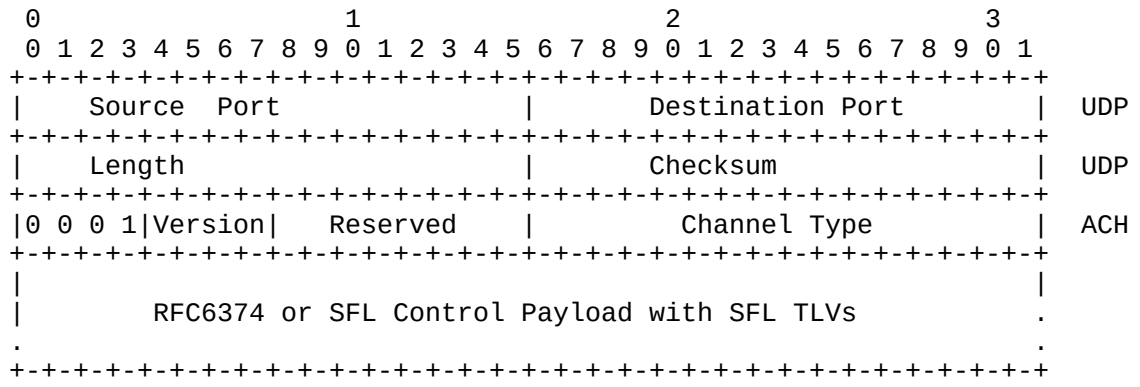


Figure 1: RFC6374 over UDP Protocol Stack

3.1. Querier to Responder

The following is rather laboured, but it is necessary to demonstrate that all of the required mapping information is carried.

Consider the direction Querier to Responder for RFC6374 Messages. The following explains the identifier mapping.

1. Destination IP address (carried in the outer IP header (not shown)). This is used to identify the targeted RFC6374 Responder to the IP network.

2. Source IP address (carried in the outer IP header (not shown)). This is used to identify the originating RFC6374 Querier to the RFC6374 Responder in order for it to construct the return IP packet.

3. UDP Source Port used by the RFC6374 Responder to direct responses to the correct Query process on the RFC6374 Querier.

4. UDP Destination Port is used by RFC6374 Querier to direct the message to the correct process on the RFC6374 Responder.

5. IP and UDP source and destination information are reversed in the usual way in the ACH Response messages from Responder back to Querier.

6. The RFC6374 Session Identifier used by both Querier and Responder to discriminate between multiple RFC6374 sessions running concurrently between the two nodes.

7. The SFL from the SFL TLV in the RFC6374 messages is used to identify the MPLS label that is being instrumented.

8. The SFL Control Protocol Session identifier used by both Querier and Responder to discriminate between multiple RFC6374 sessions running concurrently between the two nodes and used to bind the SFL control protocol session to the RFC6374 session.

Note that a node running the SFL control protocol allocates a unique SFL in response to each SFL request, and thus there is no ambiguity as to which session between which source-destination pair a particular label belongs.

Also note that there is no restriction on the use of the same SFL by many nodes since it always known which node allocated it by reference to items 1..8 in the list above.

4. Manageability Considerations

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

5. Security Considerations

Great care needs to be taken to ensure that the UDP packets defined in this document do not enter the network from unauthorised sources. This can be achieved by careful address management and the use of appropriate access control at the network’s IP entry points.

6. IANA Considerations

IANA is requested to allocate a UDP port from the user port range:

Service Name: ACH over UDP

Port Number: TBD

Descriptiopn Transport of Associated Channel Headers over UDP

Reference	This memo

7. Acknowledgements

TBD

8. Normative References

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

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

Authors' Addresses

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

George Swallow
Cisco Systems

Email: swallow@cisco.com

Siva Sivabalan
Cisco Systems

Email: msiva@cisco.com

MPLS
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2015

S. Bryant
G. Swallow
S. Sivabalan
Cisco Systems

March 2, 2015

A Control Protocol for Synonymous Flow Labels
draft-bryant-mpls-sfl-control-00

Abstract

In draft-bryant-mpls-synonymous-flow-labels the concept of MPLS synonymous flow labels (SFL) was introduced. This document describes a control protocol that runs over an associated control header to request, withdrawn and extend the lifetime of such labels.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 3, 2015.

Copyright Notice

Table of Contents

1. Introduction 2
2. Requirements Language 2
3. SFL Control 2
3.1. SFL Control Message 3
3.2. SFL Control Proceedures 6
3.2.1. Request/Grant 6
3.2.2. Refresh 7
3.2.3. Withdraw 8
3.2.4. Timer Accuracy 9
4. Return Path 9
5. Manageability Considerations 9
6. Privacy Considerations 9
7. Security Considerations 10
8. IANA Considerations 10
9. Acknowledgements 10
10. Normative References 10
Authors' Addresses 10

1. Introduction

In [draft-bryant-mpls-synonymous-flow-labels] the concept of MPLS synonymous flow labels (SFL) was introduced. This document describes a simple control protocol that runs over an associated control header to request, withdrawn and extend the lifetime of such labels. In [draft-bryant-mpls-RFC63740-over-udp] it is shown how to run this over UDP transport.

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].

3. SFL Control

This section describes the process by which the RFC6374 Querier requests SFLs, the process by which the RFC6374 Responder sends them to the Querier, and the process for managing the SFL lifetime. SFL Control Messages are carried over the SFL Control ACH. The SFL ACH is carried over a Pseudowire(PW) in place of the PW Control Word (CW), over an MPLS LSP using the GAL, or over some other mutually agreed path. Similarly the response may be returned over a PW, over a bidirectional LSP or over some other mutually agreed path. See Section 4.

3.1. SFL Control Message

The format of an SFL Control message, which follows the Associated Channel Header (ACH), is as follows:

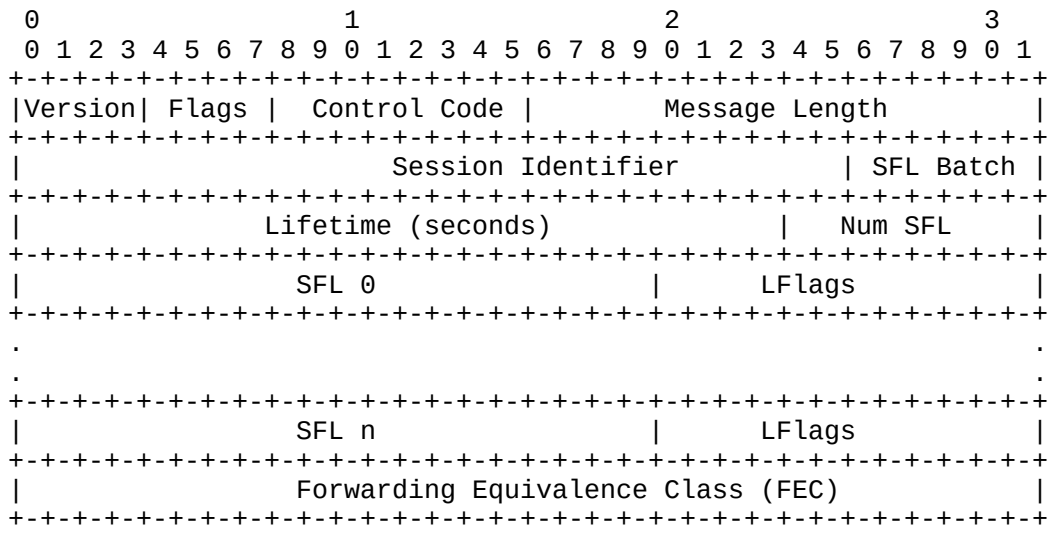


Figure 1: SFL Control Message Format

Reserved fields MUST be set to 0 and ignored upon receipt. The possible values for the remaining fields are as follows:

Version — Protocol version. Set to zero in this specification.

Flags — Message control flags.

Control Code — Code identifying the query or response type.

Message Length — Total length of this message in bytes.

Session Identifier — Set arbitrarily by the querier and used as a message handle.

SFL Batch — Used where the SFLs for this Session Identifier managed across multiple SFL Control Messages. A given set of SFLs MUST be retained in the same batch.

Lifetime — The lifetime in seconds of the SFLs in this message. In a Query message it is the requested lifetime. In a Response message it is the lifetime that the SFLs have been allocated for by the Responder. The Querier MUST

NOT use an SFL after expiry of its lifetime, a Responder MUST make the SFL available for at least its lifetime.

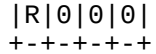
Num SFL — The number of SFLs in this SFL Batch. This MUST be constant for the lifetime of the batch.

SFL n — The n'th SFL carried in this TLV. This is an MPLS label which is a component of a label stack entry as defined in Section 2.1 of [RFC3032]. The position of a label within a batch is constant for the lifetime of the batch. Enumeration starts at zero.

LFlags — The set of flags associated with the immediately preceding SFL. See below.

FEC — The Forwarding Equivalence Class that the SFLs in this TLV correspond to. This is encoded as per Section 3.4.1 of [RFC5036].

Flags: The format of the Flags field is shown below.



SFL Control Message Flag

R: Query/Response indicator. Set to 0 for a Query and 1 for a Response.

0: Set to zero by the Sender and ignored by the Receiver.

Control Code: Set as follows according to whether the message is a Query or a Response as identified by the R flag.

For a Query:

Request This indicates that the responder is requested to allocate the set of SFLs marked with the R LFlag in this Message.

Refresh This indicates that the responder is requested to refresh the set of SFLs marked with the V LFlag in this Message.

Withdraw This indicates that the querier will no longer use the set of SFLs marked with the V Lflag and the responder may expire their lifetime.

For a Response:

Grant This indicates that the responder allocated the set of SFLs marked with the A LFlag in this Message.

Refresh-Ack This indicates that the responder has refreshed the set of SFLs marked with the V LFlag in this Message, and the lifetime is now as indicated by the lifetime field.

Withdraw-Ack This indicates that the responder has received the Withdraw message and will withdraw the SFLs

SFL-Unable The Responder was unable to satisfy the SFL Request. The details of the failure can be determined by comparing the Request and Grant messages.

Further error codes are for future study.

The LFlags field is defined as follows:

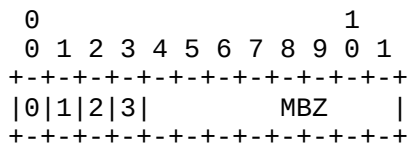


Figure 2: LFLAGS Bit Definition

Where:

0 (Valid (V)) The Label value of the corresponding SFL is valid. In an SFL Request setting the V Lflag indicates a request for the specified label value. Where an SFL has a valid flag clear in a request message this indicates that any SFL value is acceptable.

1 (Request (R)) Indicates to the Querier that this member of the SFL batch is requested. Where a value is specified in the request, but the Responder is unable honour that request, no SFL is allocated and the corresponding A flag MUST be cleared.

2 (Allocated (A) Indicates to the Querier that this SFL was allocated.

3 (Withdraw (W)) Indicates to the Responser that this SFL is to be withdrawn and to the Querier that the withdrawal has been carried out.

MBZ MUST be sent as zero and ignored on receive.

A flag value of one is true/set and a flag value of zero is false/clear. The use of these bits is described in more detail in the following sub-sections.

3.2. SFL Control Proceedures

3.2.1. Request/Grant

To request a batch of SFLs the Querier constructs an SFL Control Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. It sets the Control Message Flag to Query and the Control Code to Request. It chooses a session identifier as a handle for this transaction and as a way of binding this batch of SFLs to other operations that will use members of this SFL batch. Since members of the batch are treated as a group, the SFL Batch identifier is used to identify different SFL batches used in conjunction with the same session identifier.

The requested lifetime is set. This is the number of seconds from the time of the query to the time when the batch of SFLs will expire unless refreshed.

The Num SFL field is set to the SFL batch size.

Each SFL is set as follows: if a specific value is requested (for example for continuity across system restarts) this is written into the SFV n field and the V LFlag set. Otherwise, and including spare SFLs where an allocation is not requested, the label value is set to zero and the V LFlag is cleared. For each SFL entry where an allocation is requested the R LFlag is set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length is determined and filled in.

The Responder proceeds as follows:

It sets the control Message Flag to Response and initially sets the Control Code to Grant.

For each SFL with an R flag set, it determines whether it can honour the request, if so sets the A Lflag, and if the SFL value in the query was zero it overwrites it with the allocated SFL label value. In all other cases it leaves the SFL value and LFlag unchanged.

The lifetime field is updated with the lifetime of the SFLs if this is different from the requested lifetime.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

Where the offered lifetime is other than the requested lifetime the Querier may accept the proposed value, or withdraw the SFLs and attempt to negotiate a new set of SFLs with a different lifetime.

If the Responder is unable to allocate all of the requested SFLs it MUST respond with a response code of SFL-Unable. The Querier MUST determine whether the allocated SFLs were adequate for its purposes and MUST send a withdraw if there are not adequate. A Querier MUST NOT attempt to hoard labels in the hope that the residual labels needed may become available in the future.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before reattempting negotiation for a resource. Any failure to negotiate the required resources MUST be notified through the management interface of both Querier and Responder.

A Querier MUST NOT send an expired SFL to a Responder since to do so may invalidate another SFL operation.

3.2.2. Refresh

To request the lifetime refresh of a batch of SFLs the Querier constructs an SFL Refresh Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. It sets the Control Message Flag to Query and the Control Code to Refresh. It uses the session identifier and the SFL Batch identifier that it used to request this SFL batch.

The requested lifetime is set. This is the number of seconds from the time of the query to the time when the batch of SFLs will expire unless refreshed.

The Num SFL field is set to the SFL batch size.

Each SFL is set as follows: the allocated SFL label value is written into the SFL n field and the V LFlag set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length is determined and filled in.

The Responder proceeds as follows:

It sets the control Message Flag to Response and sets the Control Code to Refresh-Ack.

It sets the lifetime to the lifetime of the SFL.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

Where the offered lifetime is other than the requested lifetime the Querier may accept the proposed value, or withdraw the SFLs and attempt to negotiate a new set of SFLs with a different lifetime.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before reattempting negotiation for a resource. Any failure to negotiate the required resources MUST be notified through the management interface of both Querier and Responder.

3.2.3. Withdraw

To request the withdrawal of some or all of a batch of SFLs the Querier constructs an SFL Withdraw Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. It sets the Control Message Flag to Query and the Control Code to Withdraw. It uses the session identifier and the SFL Batch identifier that it used to request this SFL batch.

The requested lifetime is set to zero.

The Num SFL field is set to the SFL batch size.

Each SFL being withdrawn is set as follows: the allocated SFL label value is written into the SFL n field and the V and W LFlags set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length is determined and filled in.

The Responder proceeds as follows:

It sets the control Message Flag to Response and sets the Control Code to Withdraw-Ack.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before reattempting a Withdraw request. No more than three Withdraw requests should be made.

3.2.4. Timer Accuracy

The lifetime of SFLs is expected to be sufficiently long that there are no significant constraints on timer accuracy. A node should be conservative in its assumptions concerning the lifetime of an SFL. A Querier MUST stop using a SFL significantly before the expiry of its lifetime and a Responder must maintain an SFL in active operation significantly beyond nominal expiry. A margin of the order of minutes is RECOMMENDED.

4. Return Path

Where the LSP is a mulit-point to point, or multi-point to multi-point MPLS LSP (or other MPLS construct) the RFC6374 Address TLV MUST be included in Query packet, even if the response is requested in-band, since this is needed to provide the necessary return address for this request.

5. Manageability Considerations

This may be provided in a future version of this memo.

6. 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. Whilst the inclusion of the additional granularity does allow greater insight into the flow characteristics it does not specifically identify which node originated the packet other than by inspection of the network at the point of ingress, or inspection of the control protocol packets. This privacy threat may be mitigated by encrypting the control protocol packets, regularly

changing the synonymous labels and by concurrently using a number of such labels.

7. Security Considerations

It is assumed that this protocol is run in a well managed MPLS network with strict access controls preventing unwanted parties from generating MPLS OAM packets. The control protocol described in this memo thus introduced no additional MPLS security vulnerabilities.

8. IANA Considerations

As per the IANA considerations in [RFC5586], IANA is requested to allocate the following Channel Types in the "MPLS Generalized Associated Channel (G-ACh) Types" registry:

Value	Description	TLV Follows	Reference
0x0XXX	SFL Control	No	This

A value of 0x60 is suggested.

9. Acknowledgements

TBD

10. Normative References

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

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

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

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

Authors' Addresses

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

George Swallow
Cisco Systems

Email: swallow@cisco.com

Siva Sivabalan
Cisco Systems

Email: msiva@cisco.com

MPLS Working Group
Internet-Draft
Intended status: Informational
Expires: December 28, 2017

S. Bryant
M. Chen
Z. Li
Huawei
G. Swallow
S. Sivabalan
Cisco Systems
G. Mirsky
Ericsson

June 26, 2017

Synonymous Flow Label Framework
draft-bryant-mpls-sfl-framework-05

Abstract

draft-ietf-mpls-flow-ident describes the requirement for introducing flow identities within the MPLS architecture. This document describes a method of accomplishing this by using a technique called Synonymous Flow Labels in which labels which mimic the behaviour of other labels provide the identification service. These identifiers can be used to trigger per-flow operations on the on the packet at the receiving label switching router.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 28, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Requirements Language 2
3. Synonymous Flow Labels 3
4. User Service Traffic in the Data Plane 4
4.1. Applications Label Present 4
4.1.1. Setting TTL and the Traffic Class Bits 5
4.2. Single Label Stack 5
4.2.1. Setting TTL and the Traffic Class Bits 6
4.3. Aggregation of SFL Actions 6
5. Equal Cost Multipath Considerations 7
6. Privacy Considerations 8
7. Security Considerations 8
8. IANA Considerations 8
9. References 8
9.1. Normative References 8
9.2. Informative References 9
Authors' Addresses 9

1. Introduction

[I-D.ietf-mpls-flow-ident] describes the requirement for introducing flow identities within the MPLS architecture.

This document describes a method of accomplishing this by using a technique called Synonymous Flow Labels (SFL) (see (Section 2)) in which labels which mimic the behaviour of other labels provide the identification service. These identifiers can be used to trigger per-flow operations on the packet at the receiving label switching router.

2. Requirements Language

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

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Synonymous Flow Labels

An SFL is defined to be a label that causes exactly the same behaviour at the egress Label Switching Router (LSR) as the label it replaces, but in addition also causes an agreed action to take place on the packet. There are many possible additional actions such as the measurement of the number of received packets in a flow, triggering IPFIX inspection, triggering other types of Deep Packet Inspection, or identification of the packet source. In, for example, a Performance Monitoring (PM) application, the agreed action could be the recording of the receipt of the packet by incrementing a packet counter. This is a natural action in many MPLS implementations, and where supported this permits the implementation of high quality packet loss measurement without any change to the packet forwarding system.

Consider an MPLS application such as a pseudowire (PW), and consider that it is desired to use the approach specified in this document to make a packet loss measurement. By some method outside the scope of this text, two labels, synonymous with the PW labels are obtained from the egress terminating provider edge (T-PE). By alternating between these SFLs and using them in place of the PW label, the PW packets may be batched for counting without any impact on the PW forwarding behaviour (note that strictly only one SFL is needed in this application, but that is an optimization that is a matter for the implementor).

Now consider an MPLS application that is multi-point to point such as a VPN. Here it is necessary to identify a packet batch from a specific source. This is achieved by making the SFLs source specific, so that batches from one source are marked differently from batches from another source. The sources all operate independently and asynchronously from each other, independently co-ordinating with the destination. Each ingress is thus able to establish its own SFL to identify the sub-flow and thus enable PM per flow.

Finally we need to consider the case where there is no MPLS application label such as occurs when sending IP over an LSP. In this case introducing an SFL that was synonymous with the LSP label would introduce network wide forwarding state. This would not be acceptable for scaling reasons. We therefore have no choice but to introduce an additional label. Where penultimate hop popping (PHP) is in use, the semantics of this additional label can be similar to the LSP label. Where PHP is not in use, the semantics are similar to

an MPLS explicit NULL. In both of these cases the label has the additional semantics of the SFL.

Note that to achieve the goals set out in Section 1 SFLs need to be allocated from the platform label table.

4. User Service Traffic in the Data Plane

As noted in Section 3 it is necessary to consider two cases:

1. Applications label present

2. Single label stack

4.1. Applications Label Present

Figure 1 shows the case in which both an LSP label and an application label are present in the MPLS label stack. Traffic with no SFL function present runs over the "normal" stack, and SFL enabled flows run over the SFL stack with the SFL used to indicate the packet batch.

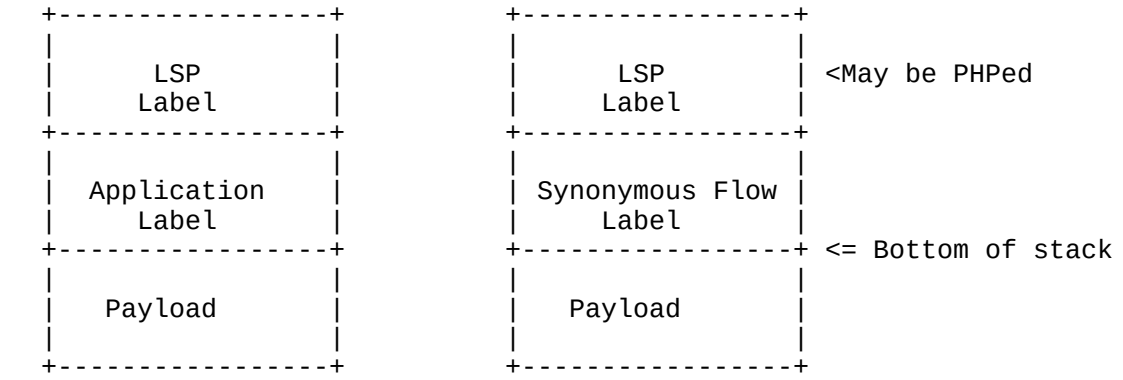


Figure 1: Use of Synonymous Labels In A Two Label MPLS Label Stack

At the egress LSR the LSP label is popped (if present). Then the SFL is processed in exactly the same way as the corresponding application label would have been processed.

4.1.1. Setting TTL and the Traffic Class Bits

The TTL and the Traffic Class bits [RFC5462] in the SFL LSE would normally be set to the same value as would have been set in the label that the SFL is synonymous with. However it is recognised that there may be an applications need to set the SFL to some other value. An example would be where it was desired to cause the SFL to trigger an action in the TTL expiry exception path as part of the label action.

4.2. Single Label Stack

Figure 2 shows the case in which only an LSP label is present in the MPLS label stack. Traffic with no SFL function present runs over the "normal" stack and SFL enabled flows run over the SFL stack with the SFL used to indicate the packet batch. However in this case it is necessary for the ingress LSR to first push the SFL and then to push the LSP label.

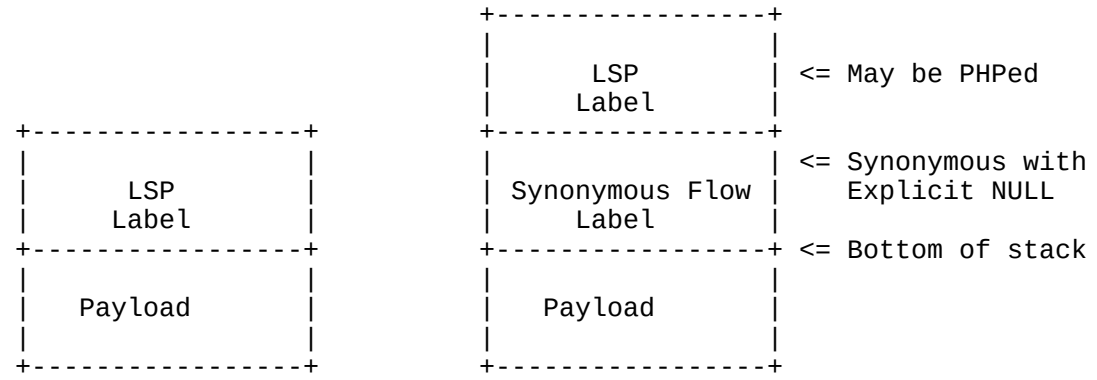


Figure 2: Use of Synonymous Labels In A Single Label MPLS Label Stack

At the receiving LSR it is necessary to consider two cases:

1. Where the LSP label is still present

2. Where the LSP label is penultimate hop popped

If the LSP label is present, it processed exactly as it would normally processed and then it is popped. This reveals the SFL which in the case of [RFC6374] measurements is simply counted and then discarded. In this respect the processing of the SFL is synonymous

with an Explicit NULL. As the SFL is the bottom of stack, the IP packet that follows is processed as normal.

If the LSP label is not present due to PHP action in the upstream LSR, two almost equivalent processing actions can take place. Either the SFL can be treated as an LSP label that was not PHPed and the additional associated SFL action is taken when the label is processed. Alternatively, it can be treated as an explicit NULL with associated SFL actions. From the perspective of the measurement system described in this document the behaviour of two approaches are indistinguishable and thus either may be implemented.

4.2.1. Setting TTL and the Traffic Class Bits

The TTL and the Traffic Class considerations described in Section 4.1.1 apply.

4.3. Aggregation of SFL Actions

There are cases where it is desirable to aggregate an SFL action against a number of labels. For example where it is desirable to have one counter record the number of packets received over a group of application labels, or where the number of labels used by a single application is large, and consequently the increase in the number of allocated labels needed to support the SFL actions consequently becomes too large to be viable. In these circumstances it would be necessary to introduce an additional label in the stack to act as an aggregate instruction. This is not strictly a synonymous action in that the SFL is not replacing a existing label, but is somewhat similar to the single label case shown in Section 4.2, and the same signalling, management and configuration tools would be applicable.

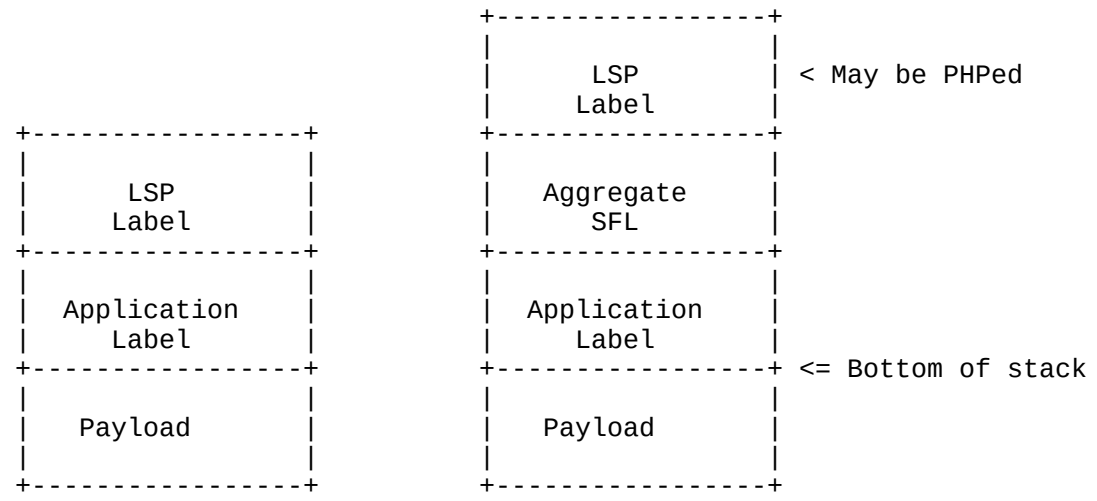


Figure 3: Aggregate SFL Actions

The Aggregate SFL is shown in the label stack depicted in Figure 3 as preceding the application label, however the choice of position before, or after, the application label will be application specific. In the case described in Section 4.1, by definition the SFL has the full application context. In this case the positioning will depend on whether the SFL action needs the full context of the application to perform its action and whether the complexity of the application will be increased by finding an SFL following the application label.

5. Equal Cost Multipath Considerations

The introduction to an SFL to an existing flow may cause that flow to take a different path through the network under conditions of Equal Cost Multipath (ECMP). This is turn may invalidate the certain uses of the SFL such as performance measurement applications. Where this is a problem there are two solutions worthy of consideration:

1. The operator can elect to always run with the SFL in place in the MPLS label stack.

2. The operator can elect to use [RFC6790] Entropy Labels in a network that fully supports this type of ECMP. If this approach is adopted, the intervening MPLS network MUST NOT load balance on any packet field other than the entropy label. Note that this is stricter than the text in Section 4.2 of [RFC6790]. In networks

in which the ECMP decision is independent of both the value of any other label in the label stack, and the MPLS payload, the path of the flow with the SFL will be congruent with the path without the SFL.

6. Privacy Considerations

Recent IETF concerns on pervasive monitoring are described in [RFC7258]. The inclusion of originating and/or flow information in a packet provides more identity information and hence potentially degrades the privacy of the communication. Whilst the inclusion of the additional granularity does allow greater insight into the flow characteristics it does not specifically identify which node originated the packet other than by inspection of the network at the point of ingress, or inspection of the control protocol packets. This privacy threat may be mitigated by encrypting the control protocol packets, regularly changing the synonymous labels and by concurrently using a number of such labels. Minimizing the scope of the identity indication can be useful in minimizing the observability of the flow characteristics.

7. Security Considerations

The issue noted in Section 6 is a security consideration. There are no other new security issues associated with the MPLS dataplane. Any control protocol used to request SFLs will need to ensure the legitimacy of the request.

8. IANA Considerations

This draft makes no IANA requests.

9. References

9.1. Normative References

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

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <http://www.rfc-editor.org/info/rfc5462>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.ietf-mpls-flow-ident] Bryant, S., Pignataro, C., Chen, M., Li, Z., and G. Mirsky, "MPLS Flow Identification Considerations", draft-ietf-mpls-flow-ident-04 (work in progress), February 2017.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <http://www.rfc-editor.org/info/rfc6374>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

Authors' Addresses

Stewart Bryant
Huawei

Email: stewart.bryant@gmail.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

George Swallow
Cisco Systems

Email: swallow@cisco.com

Siva Sivabalan
Cisco Systems

Email: msiva@cisco.com

Gregory Mirsky
Ericsson

Email: gregory.mirsky@eicsson.com

TEAS WG
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

A. Deshmukh
K. Kompella
Juniper Networks, Inc.

July 8, 2016

RSVP Extensions for RMR
draft-deshmukh-rsvp-rmr-extension-00

Abstract

Rings are the most common topology in access and aggregation networks. However, the use of MPLS as the transport protocol for rings is very limited today. draft-ietf-mpls-rmr-02 describes a mechanism to handle rings efficiently using MPLS. This document describes the extensions to the RSVP protocol for signaling MPLS label switched paths in rings.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. RSVP Extensions 3
3.1. Session Object 4
3.2. SENDER_TEMPLATE,FILTER_SPEC Objects 5
4. Ring Signaling Procedures 5
4.1. Differences from regular RSVP-TE LSPs 5
4.2. LSP signaling 5
4.3. Protection 8
4.4. Ring changes 9
4.5. Bandwidth management 10
5. Security Considerations 11
6. Contributors 11
7. IANA Considerations 12
8. References 12
8.1. Normative References 12
8.2. Informative References 12
Authors' Addresses 13

1. Introduction

This document extends RSVP-TE [RFC3209] to establish label-switched path (LSP) tunnels in the ring topology. Rings are auto-discovered using the mechanisms mentioned in the [draft-ietf-mpls-rmr-02]. Either IS-IS [RFC5305] or OSPF[RFC3630] can be used as the IGP for auto-discovering the rings.

After the rings are auto-discovered, each ring node knows its clockwise (CW) and anticlockwise (AC) ring neighbors and its ring links. All of the express links in the ring also get identified as part of the auto-discovery process. At this point, every node in the ring informs the RSVP protocol to begin the signaling of the ring LSPs.

Section 2 covers the terminology used in this document. Section 3 presents the RSVP protocol extensions needed to support MPLS rings. Section 4 describes the procedures of RSVP LSP signaling in detail.

2. Terminology

A ring consists of a subset of n nodes {R_i, 0 <= i < n}. We define the direction from node R_i to R_i+1 as "clockwise" (CW) and the reverse direction as "anti-clockwise" (AC). As there may be several rings in a graph, we number each ring with a distinct ring ID RID.

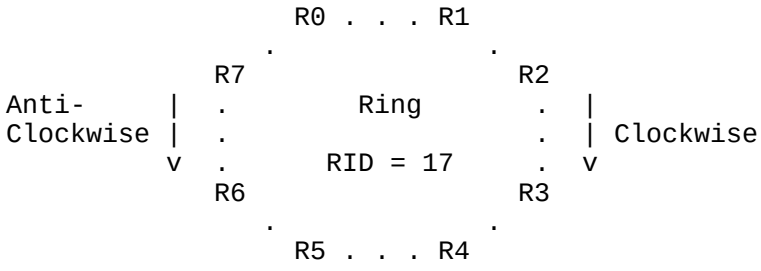


Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

Ring ID (RID): A non-zero number that identifies a ring; this is unique in some scope of a Service Provider's network. A node may belong to multiple rings.

Ring node: A member of a ring. Note that a device may belong to several rings.

Node index: A logical numbering of nodes in a ring, from zero upto one less than the ring size. Used purely for exposition in this document.

Ring neighbors: Nodes whose indices differ by one (modulo ring size).

Ring links: Links that connect ring neighbors.

Express links: Links that connect non-neighboring ring nodes.

MP2P LSP: Each LSP in the ring is a multipoint to point LSP such that LSP can have multiple ingress nodes and one egress node.

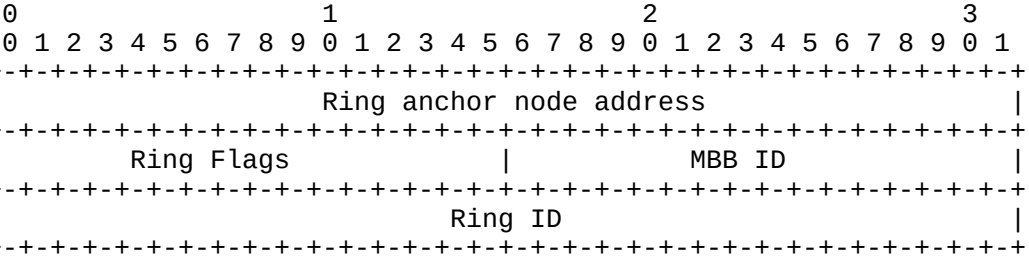
3. RSVP Extensions

Since the procedures of signaling ring LSPs will be different from the signaling of regular RSVP LSPs, a new C-Type is defined here for the SESSION object. This new C-Type will help to clearly

differentiate ring LSPs from regular LSPs. In addition, new flags are introduced in the SESSION object to represent the ring direction of the corresponding Path message.

3.1. Session Object

Class = SESSION, LSP_TUNNEL_IPv4 C-Type = TBD



SESSION Object

Ring anchor node address: IPv4 address of the anchor node. Each anchor node creates a LSP addressed to itself.

MBB ID: A 16-bit identifier used in the SESSION. This "Make-before-break" (MBB) ID is useful for graceful ring changes. If a new node is being added to the ring or some existing node goes down and we have to signal a smaller ring, in those cases, anchor node creates a new tunnel with a different "MBB ID".

Ring ID: A 32-bit number that identifies a ring; this is unique in some scope of a Service Provider’s network. This number remains constant throughout the existence of ring.

Ring Flags: For each ring, the anchor node starts signaling of a ring LSP. Ring LSP RL_i, anchored on node R_i, consists of two counter-rotating unicast LSPs that start and end at R_i. One LSP will be in the clockwise direction and other LSP will be in the anti-clockwise direction. A ring LSP is "multipoint": any node R_j can use RL_i to send traffic to R_i; this can be in either the CW or AC directions, or both (i.e., load balanced). Two new flags are defined in the SESSION object which define the ring direction of the corresponding Path message.

ClockWise(CW) Direction 0x01: This flag indicates that the corresponding Path message is traveling in the ClockWise(CW) direction along the ring.

Anti-ClockWise(AC) Direction 0x02: This flag indicates that the corresponding Path message is traveling in the Anti-ClockWise(AC) direction along the ring.

3.2. SENDER_TEMPLATE,FILTER_SPEC Objects

There will be no changes to the SENDER_TEMPLATE and FILTER_SPEC objects. The format of the above 2 objects will be similar to the definitions in RFC 3209. [RFC3209] Only the semantics of these objects will slightly change. This will be explained in section Section 4.5 below.

4. Ring Signaling Procedures

A ring node indicates in its IGP updates the ring LSP signaling protocols that it supports. This can be LDP and/or RSVP-TE. Ideally, each node should support both. If the ring is configured with RSVP as the signaling protocol, then once a ring node R_i knows the RID, its ring links and directions, it kicks off ring RSVP LSP signaling automatically.

4.1. Differences from regular RSVP-TE LSPs

Ring LSPs differ from regular RSVP-TE LSPs in several ways:

1. Ring LSPs (by construction) form a loop.

2. Ring LSPs are multipoint-to-point. Any ring node can inject traffic into a ring LSP.

3. The bandwidth of a ring LSP can change hop-to-hop.

4. Ring LSPs are protected without the use of bypass or detour LSPs. Ring LSP protection is akin to SONET/SDH ring protection.

4.2. LSP signaling

After the ring auto-discovery process, each anchor node creates a LSP addressed to itself. This ring LSP contains of a pair of counter-rotating unicast LSPs. So, for a ring containing N nodes, there will be 2N total LSPs signaled.

There is no need for ERO object in the Path message. The Path message for ring LSPs has the following format:

```
        <Path Message> ::=  <Common Header> [ <INTEGRITY> ]
                                <SESSION> <RSVP_HOP>
                                <TIME_VALUES>
                                <LABEL_REQUEST>
                                [ <SESSION_ATTRIBUTE> ]
                                <sender descriptor list>

        <sender descriptor list> ::= <sender descriptor>|
                                          <sender descriptor list> <sender descri
ptor>
        <sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>
```

The anchor node creates 2 Path messages traveling in opposite directions. The SESSION format MUST be as per the description in Section 3.1. The anchor node which creates the LSP will insert it's own address in the "Ring node anchor address" field of the SESSION object. So effectively, the Path messages are addressed to the originating node itself.

The SESSION flags of these 2 Path messages are different. The Path message sent to the CW neighbor MUST have the CW flag set in the SESSION object to signal the LSP going in the clockwise direction. The Path message sent to the AC neighbor MUST have the AC flag set to signal the LSP in the anti-clockwise direction. The details for signaling over express links will be given in a future version.

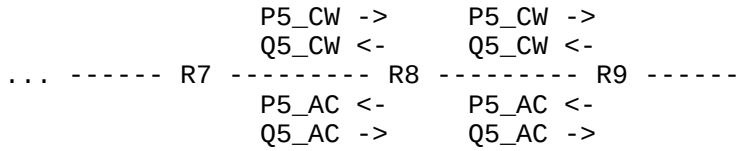
When an incoming Path message is received at the ring node R_i, it consults the results of auto-discovery to find the appropriate ring neighbor. If the incoming Path message has CW direction flag set, then R_i sends a Path message to its CW ring neighbor (and vice versa). Thus, there is no need of ERO in the Path message. The Path message is routed locally at each ring based on the ring auto-discovery calculations.

The RESV message for ring LSPs also uses the new RING_IPv4 SESSION object. When the Path message originated from the anchor node R_i reaches back to R_i, R_i generates a Resv message. Note that this means that anchor node is both Ingress and Egress for the Path message. The Resv message copies the same ring flags as received in the corresponding Path message. So, a Resv message for a CW LSP goes in the AC direction (unlike the Path message, which goes CW). This is done to correctly match Path and corresponding Resv messages at transit ring nodes. Upon receiving Resv message with CW flag set, the ring node will forward the Resv message to its AC neighbor.

Each ring node R_i allocates CW and AC labels for each ring LSP RL_k. As the signaling propagates around the ring, CW and AC labels are

exchanged. When R_i receives CW and AC labels for RL_k from its ring neighbors, primary and fast reroute (FRR) paths for RL_k are installed at R_i.

Consider the following three nodes of the ring, and their signaling interactions for LSP RL_5 originating from anchor node R5:



P corresponds to the Path message and Q corresponds to the Resv message.

Also, since ring LSPs are MP2P in nature, each ring node SHOULD also signal a Path message towards anchor node. The procedure for that is as follows:

When a ring node R5 receives a Path message initiated by anchor node R1(for anchor lsp "lsp1"), R5 SHOULD make a copy of the received Path message for "lsp1". R5 then modifies the sender-template object from the copied Path message for "lsp1". In the sender-template object, R5 uses the sender address as the loopback address of node R5 and lsp-id = X. R5 then forwards this new Path message to it’s ring neighbor. The original anchor Path message has sender address as loopback address of R1 and lsp-id = X.

So at this point, there will be 2 different path messages existing for lsp1 First Path message will be for the anchor LSP with sender address = node R1. Second Path message will be for the ring LSP with sender address = node R5.

When node R1 receives this modified Path message, it replies with the Resv message containing the same label it advertised for the original anchor lsp "lsp1". The SESSION object of the Resv message will also exactly match with the received Path message. Only the FILTER_SPEC object in the Resv message will have the sender address as loopback of node R5. As this Resv message propagates back towards R5, all the transit nodes also send the same label that they have allocated for the original anchor lsp "lsp1". So no new label routes get installed as part of signaling for this ring lsp. The anchor LSP and all of their associated ring LSPs share label routes. The label actions are described below in Section 4.3.

4.3. Protection

In the rings, there are no protection LSPs -- no node or link bypass LSPs, no standby LSPs and no detours. Protection is via the "other" direction around the ring, which is why ring LSPs are in counter-rotating pairs. Protection works in the same way for link, node and ring LSP failures.

Since each ring LSP is a MP2P LSP, any ring node can inject traffic onto a LSP whose anchor might be a different ring node. To achieve the above, an ingress route will be installed as follows at every ring node J, for a given ring-LSP with anchor Rk (say 1.2.3.4).

```
1.2.3.4  ->  (Push CL_J+1,K, NH: R_J+1)       # CW
         ->  (Push AL_J-1,K, NH: R_J-1)       # AC

          CL = Clockwise label
          AL = Anti-Clockwise label
```

Traffic will either be load balanced in the CW and AC directions or the traffic will be sent on just CW or AC lsp based on parameters such as hop-count, policy etc.

Also, 2 transit routes will be installed for the anchor LSP transiting from node Rj as follows:

```
CL_J,K ->  SWAP(CL_J+1,K,  NH: R_J+1)              #CW
       ->  SWAP(AL_J-1,K , NH: R_J-1)              #AC

           CL = Clockwise label
           AL = Anti-Clockwise label
           CW NH has weight 1, AC NH has higher-weight.


AL_J,K -> SWAP(AL_J-1,K , NH: R_J-1)  #AC
       -> SWAP(CL_J+1,K,  NH: R_J+1)  #CW

           CL = Clockwise label
           AL = Anti-Clockwise label
           AC NH has weight 1, CW NH has higher weight.
```

Suppose a packet headed in anti-clockwise direction towards R5 and it arrives at node R8. Lets say that now R8 learns there is a link

failure in the AC direction. R8 reroutes this packet back onto the clockwise direction. This reroute action is pre-programmed in the LFIB, to minimize the time between detection of a fault and the corresponding recovery action.

At this time, R8 also sends a notification to R7 that the AC direction is not working, so that R7 can similarly switch traffic to the CW direction. These notification SHOULD propagate CW until each traffic source on the ring CW of the failure uses the CW direction.For RSVP-TE, this notification is sent in the form of PathErr message.

To provide this notification, the ring node detecting failure SHOULD send a Path Error message with error code of "Notify" and an error value field of ("Tunnel locally repaired"). This Path Error code and value is same as defined in RFC 4090[RFC4090] for the notification of local repair.

Note that the failure of a node or a link will not necessarily affect all ring LSPs. Thus, it is important to identify the affected LSPs and only switch the affected LSPs.

4.4. Ring changes

A ring node can go down resulting in a smaller ring or a new node can be added to the ring which will increase the ring size. In both of the above cases, the ring auto-discovery process SHOULD kick in and it SHOULD calculate a new ring with the changed ring nodes.

When the ring auto-discovery process is complete, IGP will signal RSVP to begin the MBB process for the existing ring LSPs. For this MBB process, the anchor node will create a new Path message with a different "MBB ID" in the SESSION object. All other fields in the SESSION Object will remain same as the existing Path message(before the ring change).

This new Path message will then propagate along the ring neighbors in the same way as the original Path message. Each ring neighbor SHOULD forward the Path message to it’s appropriate neighbor based on the new auto-discovery calculations.

For the ring links which are common between the old and new LSPs, the LSPs will share resources(SE style reservation) on those ring links. Note that here we are using MBB_ID in the SESSION object to share resources instead of the LSP_ID in the SENDER_TEMPLATE Object(which is used in RSVP-TE for sharing resources as described in RFC 3209 [RFC4090]). The LSP_ID use is reserved for a different functionality as described in section Section 4.5.

4.5. Bandwidth management

For RSVP-TE LSPs, bandwidths may be signaled in both directions. However, these are not provisioned either; rather, one does "reverse call admission control". When a service needs to use an LSP, the ring node where the traffic enters the ring attempts to increase the bandwidth on the LSP to the egress. If successful, the service is admitted to the ring.

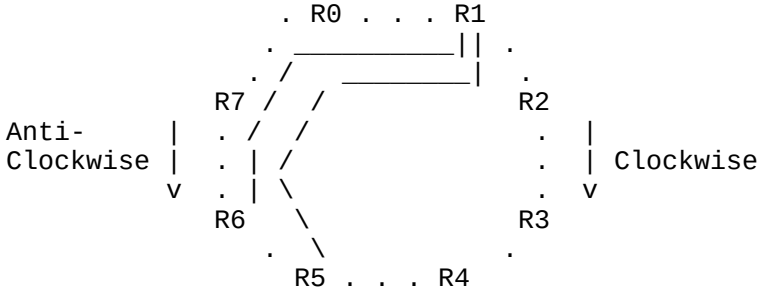


Figure 2: BW Management in Ring with 8 nodes

Let’s say that Ring node R5 wants to increase the BW for the LSP whose egress is at node R1. To achieve this BW increase, Ring node R5 has to increase BW along the LSP anchored at node R1(say lsp1).

R5 makes a copy of the existing ring Path message for lsp1. R5 then modifies the sender-template object from the copied Path message for "lsp1". In the sender-template object, R5 uses the sender address as the loopback address of node R5 and lsp-id = X+1. R5 also modifies the TSPEC object which represents the BW increase/decrease in this new Path message. R5 then forwards this new Path message to it’s ring neighbor. Note that R5 MUST also continue signaling the original anchor Path message received from ring node R1 for lsp1. The original anchor Path message has sender address as loopback address of R1.

Now, let’s say, node 5 wants to increase BW again for lsp1, then R5 adds a new SENDER_TEMPLATE object in the existing Path message for "lsp1" with sender address as loopback of node 5 and lsp-id = X+2. So at this point, there will be 2 different path messages existing for lsp1 First Path message will be for the anchor LSP with sender address = node 1. Second Path message will contain 2 SENDER_TEMPLATE objects as [node5, lsp-id = X+1] and [node5, lsp-id =X+2].

Similarly, if node R6 wants to increase the BW for "lsp1", it SHOULD create a new Path message containing SENDER_TEMPLATE object with

sender address = loopback of node 6 and lsp-id = Y+1. Thus, the LSP-ID field is local to each sender node along the ring.

If sufficient BW is available all the way towards ring node R1, then this new Path message reaches node R1. R1 generates a Resv message with the correct FILTER_SPEC object corresponding to the received SENDER_TEMPLATE object. This Resv message will also have the correct FLOWSPEC object as per the requested bandwidth.

If sufficient BW is not available at some downstream (say node R9), then ring node R9 SHOULD generate a PathErr message with the corresponding Sender Template Object. When node R5 receives this PathErr message, R5 understands that the BW increase was not successful. Note that the existing established bandwidths for lsp1 are not affected by this new PathErr message.

When ring node R5 no longer needs the BW reservation, then ring node R5 SHOULD originate a PathTear message with the appropriate Sender Template Object as described above. Every downstream node SHOULD then remove bandwidth allocated on the corresponding link on receipt of this PathTear message.

Also, note that as part of this BW increase or decrease process, any ring node does not actually change any label associated with the LSP. So, the label remains same as it was signaled initially when the anchor LSP came up.

5. 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.

6. Contributors

Ravi Singh
Juniper Networks, Inc.
1194 N Mathilda Ave
Sunnyvale, CA 94089
USA

Email: ravis@juniper.net

Santosh Esale
Juniper Networks, Inc.
1194 N Mathilda Ave
Sunnyvale, CA 94089
USA

Email: sesale@juniper.net

Raveendra Torvi
Juniper Networks, Inc.
10 Technology Park Dr
Westford, MA 01886
USA

Email: rtorvi@juniper.net

7. IANA Considerations

Requests to IANA will be made in a future version of this document.

8. References

8.1. Normative References

[I-D.ietf-mpls-rmr]
Kompella, K. and L. Contreras, "Resilient MPLS Rings", draft-ietf-mpls-rmr-02 (work in progress), July 2016.

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

8.2. Informative References

[I-D.dai-mpls-rsvp-te-mbb-label-reuse]
Dai, M. and M. Chaudhry, "MPLS RSVP-TE MBB Label Reuse", draft-dai-mpls-rsvp-te-mbb-label-reuse-01 (work in progress), September 2015.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <http://www.rfc-editor.org/info/rfc2205>.

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

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

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

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

Authors' Addresses

Abhishek Deshmukh
Juniper Networks, Inc.
10 Technology Park Dr
Westford, MA 01886
USA

Email: adeshmukh@juniper.net

Kireeti Kompella
Juniper Networks, Inc.
1194 N Mathilda Ave
Sunnyvale, CA 94089
USA

Email: kireeti@juniper.net

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2017

V. Govindan
M. Mudigonda
A. Sajassi
Cisco Systems
G. Mirsky
Ericsson

July 6, 2016

Fault Management for EVPN networks
draft-gmsm-bess-evpn-bfd-00

Abstract

This document proposes a proactive, in-band network OAM mechanism to detect loss of continuity and miss-connection faults that affect unicast and multi-destination paths, used by Broadcast, unknown Unicast and Multicast traffic, in an EVPN network. The mechanisms proposed in the draft use the principles of the widely adopted Bidirectional Forwarding Detection protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Scope of the Document 3
3. Motivation for running BFD at the network layer of EVPN . . . 3
4. Fault Detection of unicast traffic 4
5. Fault Detection of BUM traffic using ingress replication (MP2P) 5
6. Fault Detection of BUM traffic using P2MP tunnels (LSM) . . . 5
7. BFD packet encapsulation 5
7.1. Using GAL/G-ACh encapsulation without IP headers 5
7.1.1. Ingress replication 5
7.1.1.1. Alternative encapsulation format 5
7.1.2. LSM 6
7.1.3. Unicast 6
7.1.3.1. Alternative encapsulation format 6
7.2. Using IP headers 7
8. Scalability Considerations 7
9. IANA Considerations 7
10. Security Considerations 8
11. References 8
11.1. Normative References 8
11.2. Informative References 10
Authors' Addresses 10

1. Introduction

[I-D.salam-l2vpn-evpn-oam-req-frmwk] and [I-D.ooamdt-rtgwg-ooam-requirement] outlines the OAM requirements of Ethernet VPN networks [RFC7432]. This document proposes mechanisms for proactive fault detection at the network(overlay) OAM layer of EVPN. EVPN fault detection mechanisms need to consider unicast and Broadcast and unknown Unicast (BUM) traffic separately since they map to different FECs in EVPN, hence this document proposes different fault detection mechanisms to suit each type using the principles of[RFC5880],[RFC5884] and Point-to-multipoint BFD[I-D.ietf-bfd-multipoint] and [I-D.ietf-bfd-multipoint-active-tail].

1.1. Requirements Language

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

2. Scope of the Document

This document proposes proactive fault detection for EVPN [RFC7432] using BFD mechanisms for:

- Unicast traffic.
- BUM traffic using Multi-point-to-Point (MP2P) tunnels (ingress replication).
- BUM traffic using Point-to-Multipoint (P2MP) tunnels (LSM).

This document does not discuss BFD mechanisms for:

- EVPN variants like PBB-EVPN [RFC7623]. This will be addressed in future versions.
- IRB solution based on EVPN [I-D.ietf-bess-evpn-inter-subnet-forwarding]. This will be addressed in future versions.
- EVPN using other encapsulations like VxLAN, NVGRE and MPLS over GRE [I-D.ietf-bess-evpn-overlay].
- BUM traffic using MP2MP tunnels will also be addressed in a future version of this document.

This specification describes procedures only for BFD asynchronous mode. BFD demand mode is outside the scope of this specification. Further, the use of the Echo function is outside the scope of this specification.

3. Motivation for running BFD at the network layer of EVPN

The choice of running BFD at the network layer of the OAM model for EVPN [I-D.salam-l2vpn-evpn-oam-req-frmwk] and [I-D.ooamdt-rtgwg-ooam-requirement] was made after considering the following:

- In addition to detecting link failures in the EVPN network, BFD sessions at the network layer can be used to monitor the successful programming of labels used for setting up MP2P and P2MP

EVPN tunnels transporting Unicast and BUM traffic. The scope of reachability detection covers the ingress and the egress EVPN PE nodes and the network connecting them.

- o Monitoring a representative set of path(s) or a particular path among the multiple paths available between two EVPN PE nodes could be done by exercising the entropy labels when they are used. However paths that cannot be realized by entropy variations cannot be monitored. Fault monitoring requirements outlined by [I-D.salam-l2vpn-evpn-oam-req-frmwk] are addressed by the mechanisms proposed by this draft.

Successful establishment and maintenance of BFD sessions between EVPN PE nodes does not fully guarantee that the EVPN service is functioning. For example, an egress EVPN-PE can understand the EVPN label but could switch data to incorrect interface. However, once BFD sessions in the EVPN Network Layer reach UP state, it does provide additional confidence that data transported using those tunnels will reach the expected egress node. When the BFD session in EVPN overlay goes down that can be used as indication of the Loss-of-Connectivity defect in the EVPN underlay that would cause EVPN service failure.

4. Fault Detection of unicast traffic

The mechanisms specified in BFD for MPLS LSPs [RFC5884] [RFC7726] can be applied to bootstrap and maintain BFD sessions for unicast EVPN traffic. The discriminators required for de-multiplexing the BFD sessions MUST be exchanged using EVPN LSP ping specifying the Unicast EVPN FEC [I-D.jain-bess-evpn-lsp-ping] before establishing the BFD session. This is needed since the MPLS label stack does not contain enough information to disambiguate the sender of the packet. The usage of MPLS entropy labels take care of addressing the requirement of monitoring various paths of the multi-path server layer network [RFC6790]. Each unique realizable path between the participating PE routers MAY be monitored separately when entropy labels are used. The multi-path connectivity between two PE routers MUST be tracked by at least one representative BFD session, in which case the granularity of fault-detection would be coarser. The PE node receiving the EVPN LSP ping MUST allocate BFD discriminators using the procedures defined in [RFC7726]. Note that once the BFD session for the EVPN label is UP, either end of the BFD session MUST NOT change the local discriminator values of the BFD Control packets it generates, unless it first brings down the session as specified in [RFC5884].

5. Fault Detection of BUM traffic using ingress replication (MP2P)

Ingress replication uses separate MP2P tunnels for transporting BUM traffic from the ingress PE (head) to a set of one or more egress PEs (tails). The fault detection mechanism proposed by this document takes advantage of the fact that a unique copy is made by the head for each tail. Another key aspect to be considered in EVPN is the advertisement of the inclusive multicast route. The BUM traffic flows from a head node to a particular tail only after the head receives the inclusive multicast route containing the BUM EVPN label (downstream allocated) corresponding to the MP2P tunnel. The head-end PE performing ingress replication MUST initiate an EVPN LSP ping using the inclusive multicast FEC [I-D.jain-bess-evpn-lsp-ping] upon receiving an inclusive multicast route from a tail to bootstrap the BFD session. There MAY exist multiple BFD sessions between a head PE an individual tail due to the usage of entropy labels [RFC6790] for an inclusive multicast FEC. The PE node receiving the EVPN LSP ping MUST allocate BFD discriminators using the procedures defined in [RFC7726]. Note that once the BFD session for the EVPN label is UP, either end of the BFD session MUST NOT change the local discriminator values of the BFD Control packets it generates, unless it first brings down the session as specified in [RFC5884].

6. Fault Detection of BUM traffic using P2MP tunnels (LSM)

TBD.

7. BFD packet encapsulation

7.1. Using GAL/G-ACh encapsulation without IP headers

7.1.1. Ingress replication

The packet contains the following labels: LSP label (transport) when not using PHP, the optional entropy label, the BUM label and the SH label[RFC7432] (where applicable). The G-ACh type is set to TBD. The G-Ach payload of the packet MUST contain the L2 header (in overlay space) followed by the IP header encapsulating the BFD packet. The MAC address of the inner packet is used to validate the <EVI, MAC> in the receiving node. The discriminator values of BFD are obtained through negotiation through the out-of-band EVPN LSP ping.

7.1.1.1. Alternative encapsulation format

A new TLV can be defined as proposed in Sec 3 of [RFC6428] to include the EVPN FEC information as a TLV following the BFD Control packet.

The format of the TLV can be reused from the EVPN Inclusive Multicast sub-TLV proposed by Fig 2 of [I-D.jain-bess-evpn-lsp-ping].

A new type (TBD3) to indicate the EVPN Inclusive Multicast SubTLV is requested from the "CC/ CV MEP-ID TLV" registry [RFC6428].

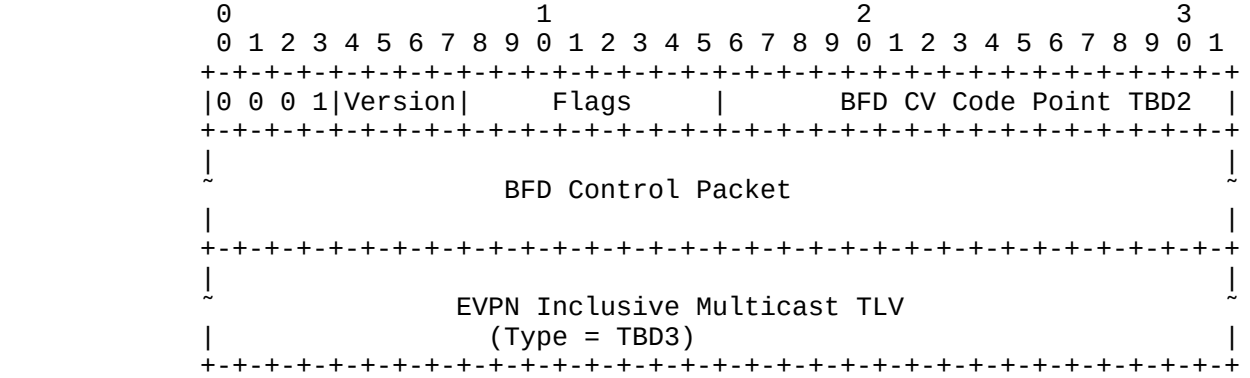


Figure 1: BFD-EVPN CV Message for EVPN Multicast (Ingress Replication)

7.1.2. LSM

TBD.

7.1.3. Unicast

The packet contains the following labels: LSP label (transport) when not using PHP, the optional entropy label and the EVPN Unicast label. The G-ACh type is set to TBD. The G-Ach payload of the packet MUST contain the L2 header (in overlay space) followed by the IP header encapsulating the BFD packet. The MAC address of the inner packet is used to validate the <EVI, MAC> in the receiving node. The discriminator values of BFD are obtained through negotiation through the out-of-band EVPN ping.

7.1.3.1. Alternative encapsulation format

A new TLV can be defined as proposed in Sec 3 of [RFC6428] to include the EVPN FEC information as a TLV following the BFD Control packet. The format of the TLV can be reused from the EVPN MAC sub-TLV proposed by Fig 1 of [I-D.jain-bess-evpn-lsp-ping]. A new type (TBD4) to indicate the EVPN MAC SubTLV is requested from the "CC/ CV MEP-ID TLV" registry [RFC6428].

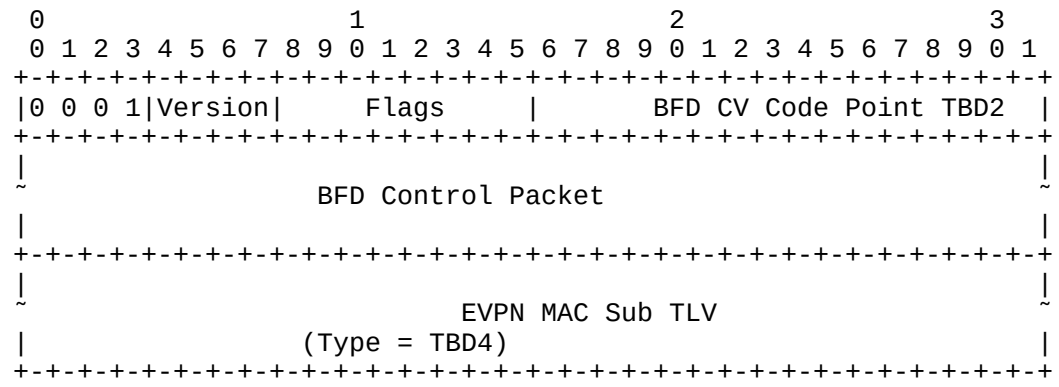


Figure 2: BFD-EVPN CV Message for EVPN Unicast

7.2. Using IP headers

The encapsulation option using IP headers will not be suited for EVPN, as using different values in the destination IP address for data and OAM (BFD) packets could cause the BFD packets to follow a different path than that of data packets. Hence this option MUST NOT be used for EVPN.

8. Scalability Considerations

The mechanisms proposed by this draft could affect the packet load on the network and its elements especially when supporting configurations involving a large number of EVIs. The option of slowing down or speeding up BFD timer values can be used by an administrator or a network management entity to maintain the overhead incurred due to fault monitoring at an acceptable level.

9. IANA Considerations

IANA is requested for two channel types from the "Pseudowire Associated Channel Types" registry in [RFC4385].

TBD1 BFD-EVPN CC message

TBD2 BFD-EVPN CV message

Ed Note: Do we need a CC code point? TBD

IANA is requested to allocate the following code-points from the "CC/CV MEP-ID TLV" registry [RFC6428]. The parent registry is the "Pseudowire Associated Channel Types" registry of [RFC4385] . All

code points within this registry shall be allocated according to the "Standards Action" procedures as specified in [RFC5226]. The items tracked in the registry will be the type, associated name, and reference. The requested values are:

TBD3 - CV code-point for BFD EVPN Inclusive multicast.

TBD4 - CV code-point for BFD EVPN Unicast.

10. Security Considerations

TBD.

11. References

11.1. Normative References

[I-D.ietf-bess-evpn-inter-subnet-forwarding]
Sajassi, A., Salam, S., Thoria, S., Rekhter, Y., Drake, J., Yong, L., and L. Dunbar, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-01 (work in progress), October 2015.

[I-D.ietf-bess-evpn-overlay]
Sajassi, A., Drake, J., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution using EVPN", draft-ietf-bess-evpn-overlay-04 (work in progress), June 2016.

[I-D.ietf-bfd-multipoint]
Katz, D., Ward, D., and J. Networks, "BFD for Multipoint Networks", draft-ietf-bfd-multipoint-08 (work in progress), April 2016.

[I-D.ietf-bfd-multipoint-active-tail]
Katz, D., Ward, D., and J. Networks, "BFD Multipoint Active Tails.", draft-ietf-bfd-multipoint-active-tail-02 (work in progress), May 2016.

[I-D.jain-bess-evpn-lsp-ping]
Jain, P., Boutros, S., and S. Salam, "LSP-Ping Mechanisms for EVPN and PBB-EVPN", draft-jain-bess-evpn-lsp-ping-03 (work in progress), May 2016.

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

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <http://www.rfc-editor.org/info/rfc4385>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://www.rfc-editor.org/info/rfc5880>.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <http://www.rfc-editor.org/info/rfc5884>.

[RFC6428] Allan, D., Ed., Swallow, G., Ed., and J. Drake, Ed., "Proactive Connectivity Verification, Continuity Check, and Remote Defect Indication for the MPLS Transport Profile", RFC 6428, DOI 10.17487/RFC6428, November 2011, <http://www.rfc-editor.org/info/rfc6428>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

[RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <http://www.rfc-editor.org/info/rfc7623>.

[RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <http://www.rfc-editor.org/info/rfc7726>.

11.2. Informative References

[I-D.ooamdt-rtgwg-ooam-requirement]
Kumar, N., Pignataro, C., Kumar, D., Mirsky, G., Chen, M., Nordmark, E., Networks, J., and D. Mozes, "Overlay OAM Requirements", draft-ooamdt-rtgwg-ooam-requirement-00 (work in progress), March 2016.

[I-D.salam-l2vpn-evpn-oam-req-frmwk]
Salam, S., Sajassi, A., Aldrin, S., and J. Drake, "E-VPN Operations, Administration and Maintenance Requirements and Framework", draft-salam-l2vpn-evpn-oam-req-frmwk-02 (work in progress), January 2014.

Authors' Addresses

Vengada Prasad Govindan
Cisco Systems

Email: venggovi@cisco.com

Mudigonda Mallik
Cisco Systems

Email: mmudigon@cisco.com

Ali Sajassi
Cisco Systems

Email: sajassi@cisco.com

Gregory Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

MPLS WG
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

K. Kompella
Juniper Networks, Inc.
L. Contreras
Telefonica

July 7, 2016

Resilient MPLS Rings
draft-ietf-mpls-rmr-02

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 is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on January 8, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Definitions 3
2. Motivation 5
3. Theory of Operation 5
3.1. Provisioning 5
3.2. Ring Nodes 6
3.3. Ring Links and Directions 6
3.3.1. Express Links 6
3.4. Ring LSPs 7
3.5. Installing Primary LFIB Entries 7
3.6. Installing FRR LFIB Entries 7
3.7. Protection 8
4. Autodiscovery 9
4.1. Overview 9
4.2. Ring Announcement Phase 11
4.3. Mastership Phase 11
4.4. Ring Identification Phase 12
4.5. Ring Changes 12
5. Ring Signaling 13
6. Ring OAM 13
7. Security Considerations 13
8. Acknowledgments 13
9. IANA Considerations 13
10. References 14
10.1. Normative References 14
10.2. Informative References 14
Authors' Addresses 14

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 [RFC5305], OSPF[RFC3630], RSVP-TE [RFC3209] and LDP [RFC5036].

The intent of this document is to handle rings that "occur naturally". Many access and aggregation networks in metros have their start as a simple ring. They may then grow into more complex topologies, for example, by adding parallel links to the ring, or by adding "express" links. The goal here is to discover these rings (with some guidance), and run MPLS over them efficiently. The intent is not to construct rings in a mesh network, and use those for protection.

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 n nodes {R_i, 0 <= i < n} of V. The directed edges {(R_i, R_i+1) and (R_i+1, R_i), 0 <= i < n-1} must be a subset of E (note that index arithmetic is done modulo n). We define the direction from node R_i to R_i+1 as "clockwise" (CW) and the reverse direction as "anticlockwise" (AC). As there may be several rings in a graph, we number each ring with a distinct ring ID RID.

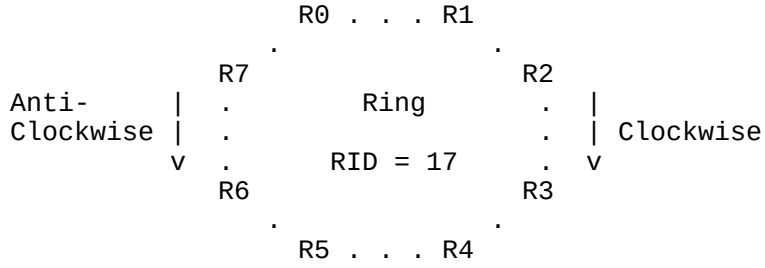


Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

Ring ID (RID): A non-zero number that identifies a ring; this is unique in some scope of a Service Provider’s network. A node may belong to multiple rings.

Ring node: A member of a ring. Note that a device may belong to several rings.

Node index: A logical numbering of nodes in a ring, from zero upto one less than the ring size. Used purely for exposition in this document.

Ring master: The ring master initiates the ring identification process. Mastership is indicated in the IGP by a two-bit field.

Ring neighbors: Nodes whose indices differ by one (modulo ring size).

Ring links: Links that connnect ring neighbors.

Express links: Links that connnect non-neighboring ring nodes.

Ring direction: A two-bit field in the IGP indicating the direction of a link. The choices are:

UN: 00 undefined link

CW: 01 clockwise ring link

AC: 10 anticlockwise ring link

EX: 11 express link

Ring Identification: The process of discovering ring nodes, ring links, link directions, and express links.

The following notation is used for ring LSPs:

R_k: A ring node with index k. R_k has AC neighbor R_(k-1) and CW neighbor R_(k+1).

RL_k: A (unicast) Ring LSP anchored on node R_k.

CL_jk: A label allocated by R_j for RL_k in the CW direction.

AL_jk: A label allocated by R_j for RL_k in the AC 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), for automatic bring-up of LSPs, better bandwidth management and for 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 ring ID RID. The ring is provisioned by choosing one or more ring masters for the ring and assigning them the RID. Other nodes in the ring may also be assigned this RID, or may be configured as "promiscuous". Ring discovery then kicks in. When each ring node knows its CW and AC ring neighbors and its ring links, and all express links have been identified, ring identification is complete.

Once ring identification is complete, each node signals one or more ring LSPs RL_i. RL_i, anchored on node R_i, consists of two counter-rotating unicast 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; this can be in either the CW or AC directions, or both (i.e., load balanced). Both of these counter-rotating LSPs are "active"; the choice of direction to send traffic to R_i is determined by policy at the node where traffic is injected into the ring. The default is to send traffic along the shortest path. Bidirectional connectivity between nodes R_i and R_j is achieved by using two different ring LSPs: R_i uses RL_j to reach R_j, and R_j uses RL_i to reach R_i.

3.1. Provisioning

The goal here is to provision rings with the absolute minimum configuration. The exposition below aims to achieve that using auto-discovery via a link-state IGP (see Section 4). Of course, auto-discovery can be overriden by configuration. For example, a link that would otherwise be classified by auto-discovery as a ring link might be configured not to be used for ring LSPs.

3.2. Ring Nodes

Ring nodes have a loopback address, and run a link-state IGP and an MPLS signaling protocol. To provision a node as a ring node for ring RID, the node is simply assigned that RID. A node may be part of several rings, and thus may be assigned several ring IDs.

To simplify ring provisioning even further, a node N may be made "promiscuous" by being assigned an RID of 0. A promiscuous node listens to RIDs in its IGP neighbors' link-state updates. For every non-zero RID N hears from a neighbor, N joins the corresponding ring by taking on that RID. In many situations, the use of promiscuous mode means that only one or two nodes in a ring needs to be provisioned; everything else is auto-discovered.

A ring node indicates in its IGP updates the ring LSP signaling protocols it supports. This can be LDP and/or RSVP-TE. Ideally, each node should support both.

3.3. Ring Links and Directions

Ring links must be MPLS-capable. They are by default unnumbered, point-to-point (from the IGP point of view) and "auto-bundled". The last attribute means that parallel links between ring neighbors are considered as a single link, without the need for explicit configuration for bundling (such as a Link Aggregation Group). Note that each component may be advertised separately in the IGP; however, signaling messages and labels across one component link apply to all components. Parallel links between a pair of ring nodes is often the result of having multiple lambdas or fibers between those nodes. RMR is primarily intended for operation at the packet layer; however, parallel links at the lambda or fiber layer result in parallel links at the packet layer.

A ring link is not provisioned as belonging to the ring; it is discovered to belong to ring RID if both its adjacent nodes belong to RID. A ring link's direction (CW or AC) is also discovered; this process is initiated by the ring's ring master. Note that the above two attributes can be overridden by provisioning if needed; it is then up to the provisioning system to maintain consistency across the ring.

3.3.1. Express Links

Express links are discovered once ring nodes, ring links and directions have been established. As defined earlier, express links are links joining non-neighboring ring nodes; often, this may be the

result of optically bypassing ring nodes. The use of express links will be described in a future version of this document.

3.4. Ring LSPs

Ring LSPs are not provisioned. Once a ring node R_i knows its RID, its ring links and directions, it kicks off ring LSP signaling automatically. R_i allocates CW and AC labels for each ring LSP RL_k. R_i also initiates the creation of RL_i. As the signaling propagates around the ring, CW and AC labels are exchanged. When R_i receives CW and AC labels for RL_k from its ring neighbors, primary and fast reroute (FRR) paths for RL_k are installed at R_i. More details are given in Section 5.

For RSVP-TE LSPs, bandwidths may be signaled in both directions. However, these are not provisioned either; rather, one does "reverse call admission control". When a service needs to use an LSP, the ring node where the traffic enters the ring attempts to increase the bandwidth on the LSP to the egress. If successful, the service is admitted to the ring.

3.5. Installing Primary LFIB Entries

In setting up RL_k, a node R_j sends out two labels: CL_jk to R_j-1 and AL_jk to R_j+1. R_j also receives two labels: CL_j+1,k from R_j+1, and AL_j-1,k from R_j-1. R_j can now set up the forwarding entries for RL_k. In the CW direction, R_j swaps incoming label CL_jk with CL_j+1,k with next hop R_j+1; these allow R_j to act as LSR for RL_k. R_j also installs an LFIB entry to push CL_j+1,k with next hop R_j+1 to act as ingress for RL_k. Similarly, in the AC direction, R_j swaps incoming label AL_jk with AL_j-1,k with next hop R_j-1 (as LSR), and an entry to push AL_j-1,k with next hop R_j-1 (as ingress).

Clearly, R_k does not act as ingress for its own LSPs. However, if these LSPs use UHP, then R_k installs LFIB entries to pop CL_k,k for packets received from R_k-1 and to pop AL_k,k for packets received from R_k+1.

3.6. Installing FRR LFIB Entries

At the same time that R_j sets up its primary CW and AC LFIB entries, it can also set up the protection forwarding entries for RL_k. In the CW direction, R_j sets up an FRR LFIB entry to swap incoming label CL_jk with AL_j-1,k with next hop R_j-1. In the AC direction, R_j sets up an FRR LFIB entry to swap incoming label AL_jk with CL_j+1,k with next hop R_j+1. Again, R_k does not install FRR LFIB entries in this manner.

3.7. Protection

In this scheme, there are no protection LSPs as such -- no node or link bypass LSPs, no standby LSPs, no detours, and no LFA-type protection. Protection is via the "other" direction around the ring, which is why ring LSPs are in counter-rotating pairs. Protection works in the same way for link, node and ring LSP failures.

If a node R_j detects a failure from R_j+1 -- either all links to R_j+1 fail, or R_j+1 itself fails, R_j switches traffic on all CW ring LSPs to the AC direction using the FRR LFIB entries. If the failure is specific to a single ring LSP, R_j switches traffic just for that LSP. In either case, this switchover can be very fast, as the FRR LFIB entries can be preprogrammed. Fast detection and fast switchover lead to minimal traffic loss.

R_j then sends an indication to R_j-1 that the CW direction is not working, so that R_j-1 can similarly switch traffic to the AC direction. For RSVP-TE, this indication can be a PathErr or a Notify; other signaling protocols have similar indications. These indications propagate AC until each traffic source on the ring AC of the failure uses the AC 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.

Note that the failure of a node or a link will not necessarily affect all ring LSPs. Thus, it is important to identify the affected LSPs (and switch them), but to leave the rest alone.

One point to note is that when a ring node, say R_j, fails, RL_j is clearly unusable. However, the above protection scheme will cause a traffic loop: R_j-1 detects a failure CW, and protects by sending CW traffic on RL_j back all the way to R_j+1, which in turn sends traffic to R_j-1, etc. There are three proposals to avoid this:

1. Each ring node acting as ingress sends traffic with a TTL of at most 2*n, where n is the number of nodes in the ring.

2. A ring node sends protected traffic (i.e., traffic switched from CW to AC or vice versa) with TTL just large enough to reach the egress.

3. A ring node sends protected traffic with a special purpose label below the ring LSP label. A protecting node first checks for the presence of this label; if present, it means that the traffic is looping and MUST be dropped.

It is recommended that (2) be implemented. The other methods are optional.

4. Autodiscovery

4.1. Overview

Auto-discovery proceeds in three phases. The first phase is the announcement phase. The second phase is the mastership phase. The third phase is the ring identification phase.

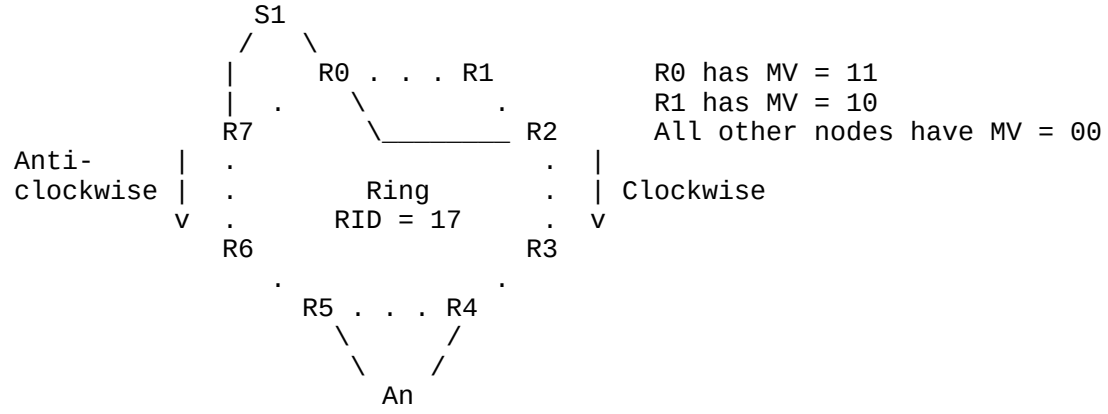
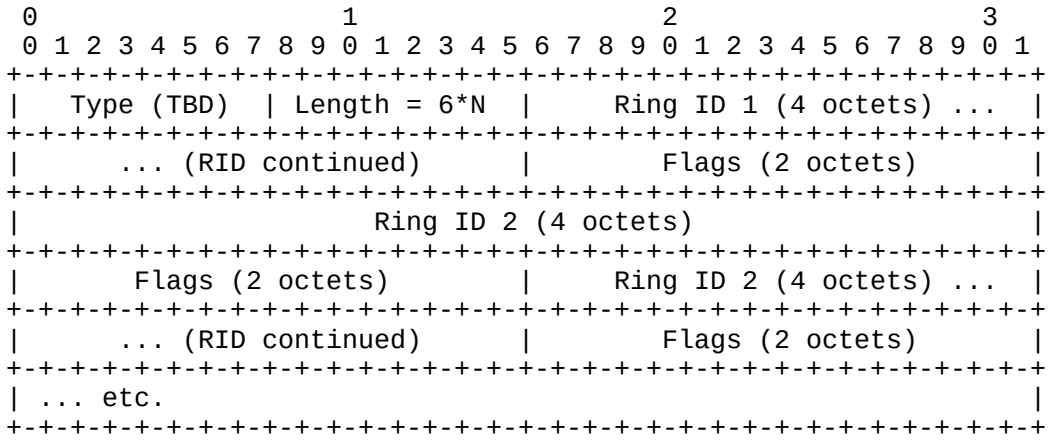
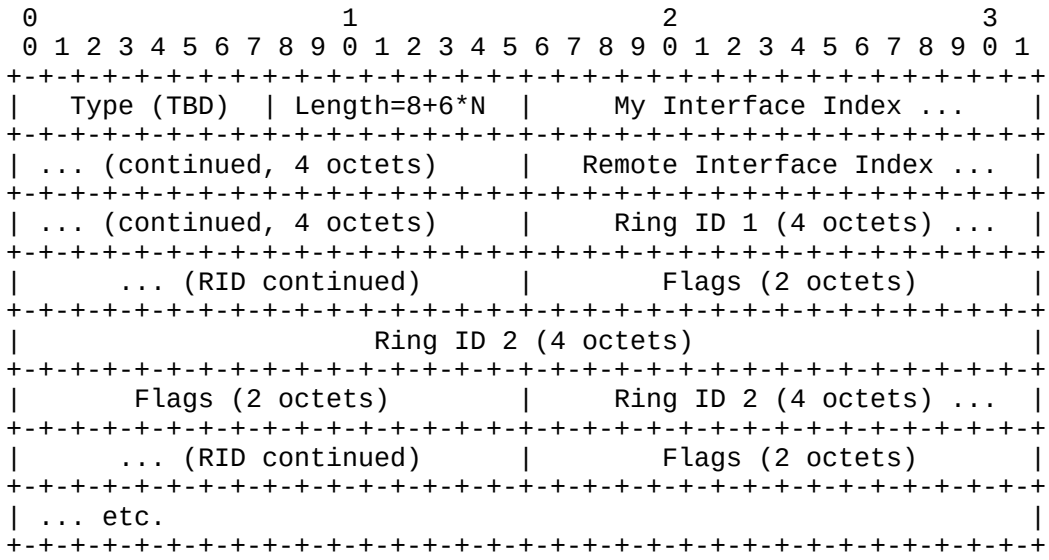


Figure 2: Ring with non-ring nodes and links

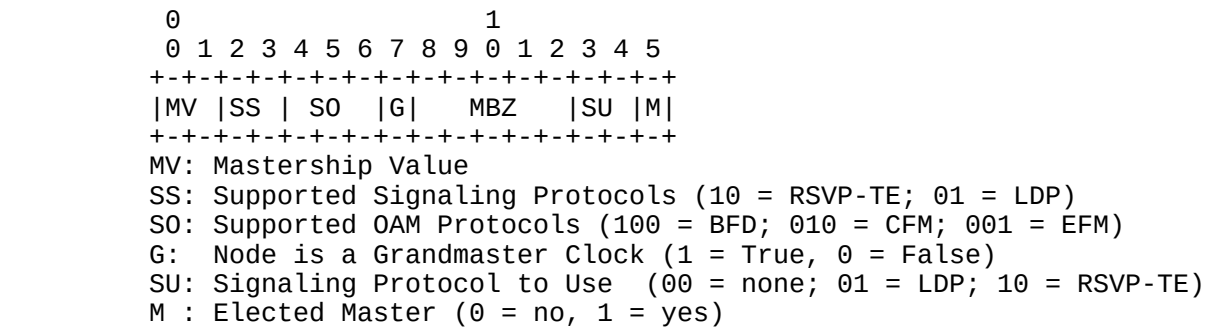
In what follows, we refer to a ring node and a rink link Type-Length-Value (TLV). These are new TLVs that contain RIDs and associated flags. A ring node TLV is a TLV that contains information for each ring that this node participates in. A ring link TLV identifies a link and contains information about every ring that that link is part of.



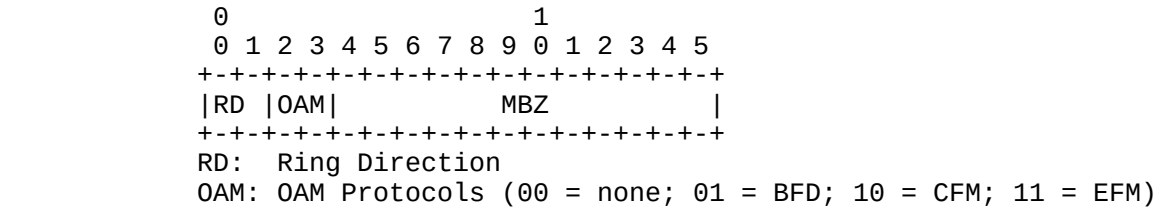
Ring Node TLV Format



Ring Link TLV Format



Flags for a Ring Node TLV



Flags for a Ring Link TLV

4.2. Ring Announcement Phase

Each node participating in an MPLS ring is assigned an RID; in the example, RID = 17. A node is also provisioned with a mastership value. Each node advertises a ring node TLV for each ring it is participating in, along with the associated flags. It then starts timer T1.

A node in promiscuous mode doesn't advertise any ring node TLVs. However, when it hears a ring node TLV from an IGP neighbor, it joins that ring, and sends its own ring node TLV with that RID.

The announcement phase allows a ring node to discover other ring nodes in the same ring so that a ring master can be elected.

4.3. Mastership Phase

When timer T1 fires, a node enters the mastership phase. In this phase, each ring node N starts timer T2 and checks if it is master. If it is the node with the lowest loopback address of all nodes with the highest mastership values, N declares itself master by readvertising its ring node TLV with the M bit set.

When timer T2 fires, each node examines the ring node TLVs from all other nodes in the ring to identify the ring master. There should be exaclty one; if not, each node restarts timer T2 and tries again. The nodes that set their M bit should be extra careful in advertising their M bit in subsequent tries.

4.4. Ring Identification Phase

When there is exactly one ring master M, M enters the Ring Identification Phase. M indicates that it has successfully completed this phase by advertising ring link TLVs. This is the trigger for M's CW neighbor to enter the Ring Identification Phase. This phase passes CW until all ring nodes have completed ring identification.

In the Ring Identification Phase, a node X that has two or more IGP neighbors that belong to the ring picks one of them to be its CW ring neighbor. If X is the ring master, it also picks a node as its AC ring neighbor. If there are exactly two such nodes, this step is trivial. If not, X computes a ring that includes all nodes that have completed the Ring Identification Phase (as seen by their ring link TLVs) and further contains the maximal number of nodes that belong to the ring. Based on that, X picks a CW neighbor and inserts ring link TLVs with ring direction CW for each link to its CW neighbor; X also inserts a ring link TLV with direction AC for each link to its AC neighbor. Then, X determines its express links. These are links connected to ring nodes that are not ring neighbors. X advertises ring link TLVs for express links by setting the link direction to "express link".

4.5. Ring Changes

The main changes to a ring are:

ring link addition;

ring link deletion;

ring node addition; and

ring node deletion.

The main goal of handling ring changes is (as much as possible) not to perturb existing ring operation. Thus, if the ring master hasn't changed, all of the above changes should be local to the point of change. Link adds just update the IGP; signaling should take advantage of the new capacity as soon as it learns. Link deletions in the case of parallel links also show up as a change in capacity (until the last link in the bundle is removed.)

The removal of the last ring link between two nodes, or the removal of a ring node is an event that triggers protection switching. In a simple ring, the result is a broken ring. However, if a ring has express links, then it may be able to converge to a smaller ring with protection. Details of this process will be given in a future version.

The addition of a new ring node can also be handled incrementally. Again, the details of this process will be given in a futre version.

5. Ring Signaling

A future version of this document will specify protocol-independent details about ring LSP signaling.

6. Ring OAM

Each ring node should advertise in its ring node TLV the OAM protocols it supports. Each ring node is expected to run a link-level OAM over each ring link. This should be an OAM protocol that both neighbors agree on. The default hello time is 3.3 millisecond.

Each ring node also sends OAM messages over each direction of its ring LSP. This is a multi-hop OAM to check LSP liveness; typically, BFD would be used for this. The node chooses the hello interval; the default is once a second.

7. 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.

8. Acknowledgments

Many thanks to Pierre Bichon whose exemplar of self-organizing networks and whose urging for ever simpler provisioning led to the notion of promiscuous nodes.

9. IANA Considerations

There are no requests as yet to IANA for this document.

10. References

10.1. Normative References

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

10.2. Informative References

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

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

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <http://www.rfc-editor.org/info/rfc5036>.

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

Authors' Addresses

Kireeti Kompella
Juniper Networks, Inc.
1133 Innovation Drive
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

Luis M. Contreras
Telefonica
Ronda de la Comunicacion
Sur-3 building, 3rd floor
Madrid 28050
Spain

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

spring
Internet-Draft
Intended status: Informational
Expires: December 24, 2016

R. Leipnitz, Ed.
R. Geib
Deutsche Telekom
June 22, 2016

A scalable and topology aware MPLS data plane monitoring system draft-leipnitz-spring-pms-implementation-report-00

Abstract

This document reports round-trip delay measurements captured by a single MPLS Path Monitoring System (PMS) compared with results of an IPPM conformant measurement system, consisting of three different Measurement Agents. The measurements were made in a research backbone with an LDP control plane. The packets of the MPLS PMS use label stacks similar to those to be used by a segment routing MPLS PMS. The measurement packets of the MPLS PMS remained in the network data plane.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 24, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
2. Measurement system implementation 3
2.1. A PMS based round-trip delay measurement system 3
2.2. Perfas+ IPPM measurement system 4
3. Test set up 4
4. Measurement Result Evaluation 6
5. Measurement results 6
5.1. Round-trip delay measurement and ADK test results 6
5.2. PMS delay measurements with IP-address variation 9
6. Error Calibration 10
7. Summary 11
8. Acknowledgements 12
9. IANA Considerations 12
10. Security Considerations 12
11. References 12
11.1. Normative References 12
11.2. Informative References 13
Appendix A. ADK2 Test Source Code 13
Authors' Addresses 22

1. Introduction

Deutsche Telekom has implemented an MPLS Path Monitoring System (PMS). The PMS operates on MPLS networks with LDP control plane. Forwarding follows the principles of Segment Routing, i.e. the packets sent by the PMS use stacked transport labels to execute a combination of MPLS paths and finally return to the PMS. The PMS is connected to a research backbone of Deutsche Telekom spanning parts of Germany. One of the new network monitoring features enabled by Segment Routing are round-trip delay measurements purely executed in data plane. Deutsche Telekom captured delays between three IPPM standard conformant Measurement Agents and compared these with delays measured along identical backbone paths by a single PMS. To prove that the same delays were measured the IPPM results were then compared with the PMS results by applying IPPM methodology as specified by [RFC6576]. Some results passed this test, while others did not. The results of both systems seemed to differ by very small and relatively stable latencies. As the research network only offered single paths between the involved routers, processing of different flows in parallel forwarding instances of the routers along the paths offered an explanation. The PMS was used to execute some

measurements whose results at least are not contradicting that assumption.

The results reported here show that a PMS [I-D.ietf-spring-oam-usecase] can be built and operated (also as part of an LDP based MPLS network). To set up packets with proper label stacks, the PMS needs to be aware of the MPLS topology of the network. MPLS topology awareness within an LDP based network requires reasonable effort. Segment Routing will significantly simplify detection of the MPLS topology. Delay measurements where picked here to give an example of a feature which can be supported by a PMS. Others are possible, like checking continuity of arbitrary segmented routed MPLS paths [I-D.ietf-spring-oam-usecase].

The remaining document is organized as follows: Section 2 briefly informs about the PMS and IPPM measurement system implementation. Section 3 introduces the measurement set up within the research network. Section 4 briefly discusses the test by which the measurements were compared. Section 5 informs about the test results and Section 6 about an IPPM error calibration. Section 7 sums up the document.

2. Measurement system implementation

Deutsche Telekom operates an IPPM standard conformant performance measurement system called Perfas+. Deutsche Telekom intends deployment of an MPLS PMS to monitor the IP performance in network segments connecting roughly 1000 edge routers to the IP-backbone. 11 MPLS PMS are supposed to execute backbone to edge performance monitoring. Had the monitoring system been based on IPPM, one IPPM system had been required per edge router.

2.1. A PMS based round-trip delay measurement system

Deutsche Telekom has implemented an MPLS PMS. The PMS is part of an MPLS research and development backbone of Deutsche Telekom. This backbone only supports LDP routing. The PMS works with an LDP control plane. Detecting the MPLS topology of an LDP based MPLS network is more complex, than doing this by Segment Routing. The PMS consists of the following logical components:

- An MPLS Label detection system. It is collecting MPLS routing information from all MPLS routers of the MPLS network by management plane access (see e.g. [LDP-TE], [BCP-TX])

- An MPLS topology database.

- A measurement system able to compose packets executing any combination MPLS Label Switched Paths (MPLS LSP) which are part of the MPLS topology database. The measurement system further is able to measure delays, if the final address information of the measurement packet directs the packet back to the PMS after the MPLS LSPs to be measured have been passed.

- An IGP topology detection system. It is passively listening to IGP routing.

- A measurement system which is complying to [RFC4379].

Note that the final two MPLS PMS functionalities are required if ECMP routed paths should be detected and addressed by [RFC4379] functions. No ECMP routed paths are present between the sites involved in the measurement set up. The role of these components is reduced to detection of operational issues, should the measurement not work as expected.

While the control plane of the network monitored by the PMS is LDP based, the measurement packets used to execute MPLS LSPs apply the forwarding mechanisms as within a Segment Routing network.

2.2. Perfas+ IPPM measurement system

IPPM conformant one-way delay measurements were performed by Perfas+ Measurement Agents. Three Perfas+ Measurement Agents are connected to edge routers at three different sites of the research network. Perfas+ is one of the few IPPM implementations with proven conformance to some standard IPPM metrics, like one-way delay [RFC6808]. Two of the Perfas+ Measurement Agents were synchronized by NTP only. Due to this restriction, the comparison with the PMS measurements are limited to round-trip times (round-trip delays, RTD). As no ECMP routed paths are active between the sites used for test execution, two back and forth Perfas+ one-way delay measurements between two sites were added to result in an RTD value.

3. Test set up

The test set up is shown in the figure below. The PMS and Perfas+ Measurement Agent 1 (PerfMA 1) are connected to the same LER.

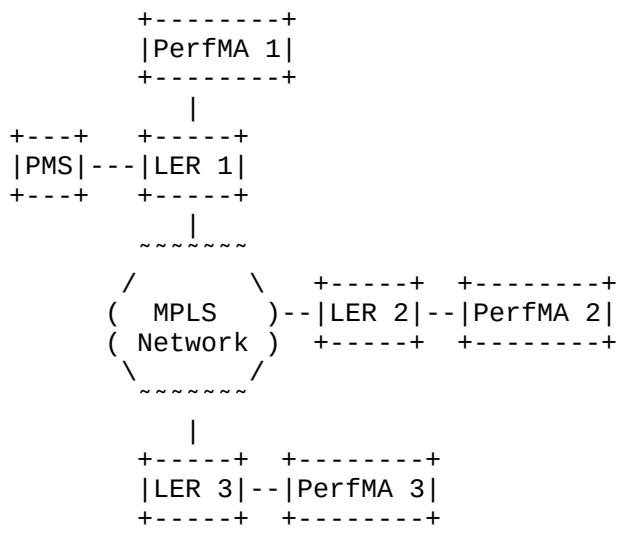


Figure 1: Test set up

The Perfas+ Measurement Agents (MAs) measure the one-way delay to each of the remote Perfas+ MAs. The PMS measures the round-trip delay from LER 1 to LER 2 and back as well the round-trip delay from LER 1 to LER 3 and back. The measurements start and terminate at the PMS, but this segment is omitted here. The round-trip delay from LER 2 to LER 3 is measured along two path combinations by the PMS. The first measurement path is LER 1 to LER 2 to LER 3 and back exactly that way. The round-trip delay LER 1 to LER 2 captured earlier by the PMS is subtracted from the result. The other measurement is LER 1 to LER 3 to LER 2 and back exactly that way. Here, the PMS round-trip delay LER 1 to LER 3 is subtracted to receive the round-trip delay LER 2 to LER 3.

There is a small LAN section causing limited additional latencies for the IPPM measurement. The measurements were executed with an IP packet size of 64 Byte. Perfas is attached by an IP-VPN. The PMS label stack is differing slightly. The assumption is that both differences have minor impact. Note that IPPM metrics expect similar results if differences in measurement set up can be neglected. The sending interval is 10 seconds periodic. A measurement mean is calculated from 10 consecutive measurement packets. The measurements were repeated for 8 hours, resulting in 288 mean values collected per round-trip delay measurement path and measurement system.

The resulting round-trip delays are divided by two and indicate the one-way delay. This seems sound, as there is no path diversity in

the research network and the low standard deviation of the results (single digit [us] figures in all cases, see test results below) indicate that no link was congested.

4. Measurement Result Evaluation

IPPM WG applies the Anderson-Darling-K-Sample (ADK) test to compare up to which temporal resolution the results of two measurements share the same statistical distribution [RFC6576]. To decide, whether Perfas+ and the PMS were measuring identical data, the round-trip delays captured along identical measurement paths were compared by an ADK test. (The ADK test source code is given at Appendix A). Note that the ADK test does not judge accuracy (i.e. it does not test whether the result is close to the true value?), ADK rather judges precision (that the test estimates whether the same value was measured by repeated measurements). As applied here, an RTD sample of Perfas+ was compared with one of the PMS captured along the same path.

To illustrate, how sensible the ADK test is to changes in a measurement environment, a PMS round-trip delay test was set up where all configurations were identical and only packet size was variable. Obviously all paths are identical, so any difference in results is caused by the packet size only (64, 128 and 256 Byte were picked). The ADK test indicated a reasonably high probability that results do not follow the same distribution in roughly half of the cases (i.e. ADK test said that the distribution of round-trip delays captured with packet size of 64 bytes follows a different distribution than the round-trip delays captured with a packet size of 128 Byte).

5. Measurement results

5.1. Round-trip delay measurement and ADK test results

The one-way delays between Perfas MA 1 and Perfas MA 2 calculated on basis of the round-trip Delay and the ADK test results comparing them to the measurement results captured by the PMS are shown in Table 1.

Test metric	PERFAS+	PMS
minimum [us]	691.5	695.5
maximum [us]	701	704.5
mean [us]	695.4	699.6
median [us]	695.5	699.5
standard deviation [us]	1.4	1.7
ADK value		278.445
ADK value with adjustment of mean		1.701
ADK value with adjustment of median		1.982

Perfas+ and PMS OWD measurement results for path LER 1 to LER 2 and ADK test results

Table 1: Perfas+ and PMS OWD measurement results for path LER 1 to LER 2 and ADK test results

The ADK test result is surprisingly good and was not expected a priori. As mentioned, ADK is a very sensible test. When IPPM WG worked on [RFC6808], the packets used by two different IPPM implementation only passed ADK after a network emulator was inserted into the measurement path. As IPPM puts more emphasis on precision than on accuracy, correcting tests samples to result by the same mean for small and constant differences is plausible. Still, the smallest temporal resolution of the standard deviation by which ADK was passed when used to compare two IPPM implementations for [RFC6808] was single digit milliseconds. No network emulator has been used when comparing Perfas+ and the PMS. After adjusting the means, ADK is passed by a temporal resolution of the standard deviation of single digit microseconds!

The one-way delays between Perfas MA 1 and Perfas MA 3 calculated on basis of the round-trip Delay and the ADK test results comparing them to the measurement results as captured by the PMS are shown in Table 2.

Test metric	PERFAS+	PMS
minimum [us]	2991.5	2983
maximum [us]	3008.5	2994.5
mean [us]	2995.7	2988.1
median [us]	2995.5	2988
standard deviation [us]	1.9	2.1
ADK value		231.638
ADK value with adjustment of mean		1.886
ADK value with adjustment of median		2.026

Perfas+ and PMS OWD measurement results for path LER 1 to LER 3 and ADK test results

Table 2: Perfas+ and PMS OWD measurement results for path LER 1 to LER 3 and ADK test results

After adjustment of the means values, also here the ADK test is passed. Comparing Table 1 with Table 2 readers figure can see, that once mean the one-way delay measured by Perfas+ is lower, while in the other case the mean one-way delay captured by the PMS is lower. This behavior was visible in all our measurements. The delays measured per path by one system were always bigger than that of the other along the same path (for all single 10 sample mean values of the time series).

We now compare the one-way delays between Perfas MA 2 and Perfas MA 3 calculated on basis of the round-trip delay and the ADK test results comparing them to the measurement results as captured by the PMS are shown in Table 3.

Test metric	PERFAS+	PMS over LER 2	PMS over LER 3
minimum [us]	3606.5	3551	3542.5
maximum [us]	3659	3568	3558
mean [us]	3611.9	3560.1	3549,8
median [us]	3609	3560	3549,5
standard deviation [us]	8.3	2.9	2.9
ADK value		231.144	231.094
ADK value with adjustment of mean		54.591	56.589
ADK value with adjustment of median		8.915	10.054

Perfas+ and PMS OWD measurement results for path LER 2 to LER 3 and ADK test results

Table 3: Perfas+ and PMS OWD measurement results for path LER 2 to LER 3 and ADK test results

In this case, the ADK test fails (the cause is the difference of the standard deviation, not the mean or median difference). Note that in terms of mean values the difference is around 50 us between Perfas and PMS. The relative error is 1,75%. While ADK indicates that both distributions deviate, human perception may confirm that both results capture delays along the same path.

It is interesting however, that the two PMS measurements deviate in the mean values. And again, the one showing the lower delay does so sample mean measurements. A brief test investigating this symptom was performed. Test and results follow in the next section.

5.2. PMS delay measurements with IP-address variation

The PMS allows to send measurement packets with different destination IP-addresses (routing based on IP-addresses only occurs from LER 1 to PMS and only in this direction). While the IP-address varied, the MPLS Label stack and thus the MPLS path was kept identical. This measurement can only be configured by CLI configuration. Per IP destination address, the mean-value of 10 round-trip delay times was captured. After some measurements the IP-addresses showing the biggest round-trip delay difference were selected for further testing. With these IP-addresses, the test was repeated at different days and daytimes. Overall we had at least 10 more measurement values of every of these IP-addresses. The PMS is connected with two interfaces to two different LERs of the same site. Both interfaces

and LERs respectively were used to perform the measurements. As has been mentioned already, the network does not have ECMP-paths. Table 4 shows the results of the two measurements with the biggest difference in results. The mean delays measured with IP-address a.b.c.0 were the smallest. They were always smaller than those delays captured with IP-address a.b.c.32, which were the biggest. The difference of the mean values from the measurement over the first interface was 19.5 us and 14.4 us over the second interface.

Interface / IP-address	mean [us]	median [us]
one / a.b.c.0	1413.2	1412
one / a.b.c.32	1432.7	1433
two / a.b.c.0	1446.4	1446
two / a.b.c.32	1460.8	1460.5

Table 4: Destination-IP-address variation

Parallel hardware processing within some or all of the routers passed on the measurement paths may be a plausible explanation. Investigating the cause for this behavior was however not the main aim of the test activities documented here. Further activities related to this issue are left to interested research.

6. Error Calibration

Section 3.7. and following of [RFC2679] recommend an error calibration of the (IPPM) measurement clients. The one-way delay of a back-to-back connection of two PERFAS+ clients is measured. Table 5 shows the characteristics of this calibration measurement. The negative values for the one-way delay shown in the table, are physically impossible. The standard deviation is very high. It was decided to calibrate with the round-trip delay which is shown in Table 6. Referring to section 3.7.3 of [RFC2679] there is a systematic error and a random error. The systematic error is the median of the measurement with 49.5 us. The random error is the difference between the median and the 2.5% percentile, which is 17 us. (The random error is the larger absolute value between the median and the 2.5% percentile and the 97.5% percentile; the calculation is |49.5 - 32.5| > |49.5 - 59.5|). The resolution of the PERFAS+ Measurement Agents is 1 us, so the absolute random error is 19 us. So measurement error is 49.5 +/- 19 us. (The synchronization error is 0, as two one-way delays are added, making this error disappear). There was no possibility to calibrate the PMS. The error is assumed to be the same like that of PERAS+, because the PMS is based on the same hardware (and possibly the same host-system).

Test metric	PERFAS+
minimum [us]	-55
maximum [us]	39
mean [us]	-38
median [us]	-23.1
standard deviation [us]	29.4

Table 5: measurement results of one-way delay of back-to-back connection from two PERFAS+ clients at 64 Bytes

Test metric	PERFAS+
minimum [us]	26
maximum [us]	205
mean [us]	49.1
median [us]	49.5
standard deviation [us]	7.6
2.5% percentile [us]	32.5
97.5% percentile [us]	59.5

Table 6: measurement results of both one-way delays of back-to-back connection between two PERFAS+ clients at 64 Bytes

7. Summary

By an IPPM measurement system like PERFAS+ three physical measurement clients are needed to measure the round-trip delay between all sites. With the PMS the same measurements can be performed with only one client. In theory one PMS could monitor a whole MPLS-enabled backbone. The GPS receivers of two IPPM measurement agents were not available, hence the one-way delay could not be captured with the IPPM system PERFAS+. Otherwise a direct comparison with calculated one-way delay values based on the PMS measured values would have been possible. This could be done in future. The results shown in Section 4 indicate, that the PMS measurements equal those captured by an IPPM conformant measurement system. The ADK test is successful by comparing the measurement values of the round-trip delays for packets with a size of 64 bytes. The network does not include an impairment generator (which was required within a test set up to compare independent IPPM implementations, see [RFC6808]). An impairment generator as part of the test set up will have a positive effect on the measurements and the measurements with bigger packet size will also succeed at a temporal resolution above [us] level.

8. Acknowledgements

Joachim Mende, Marc Wieland, Ralf Widera and Jens Wyduba helped to implement and operate the LDP PMS in our research network. In memoriam of Holger Zarwel, who gave our project unconditional support.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

A PMS monitoring packet should never leave the domain where it originated. It therefore should never use stale MPLS or IGP routing information. If the Label Switch Path is broken, a packet with the destination address 127.0.0.0/26 should not be routed, it should be discarded. The PMS must be configured with a measurement interval (or sum of all measurement stream intervals) that does not overload the network. Too many measurement streams with a big packet size could overload a link.

11. References

11.1. Normative References

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, DOI 10.17487/RFC2679, September 1999, <http://www.rfc-editor.org/info/rfc2679>.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <http://www.rfc-editor.org/info/rfc6576>.

[RFC6808] Ciavattone, L., Geib, R., Morton, A., and M. Wieser, "Test Plan and Results Supporting Advancement of RFC 2679 on the Standards Track", RFC 6808, DOI 10.17487/RFC6808, December 2012, <http://www.rfc-editor.org/info/rfc6808>.

11.2. Informative References

[BCP-TX] NANOG, "Best Practices for Determining Traffic Matrices in IP Networks V 4.0", 2008.

[I-D.ietf-spring-oam-usecase] Geib, R., Filsfils, C., Pignataro, C., and N. Kumar, "A Scalable and Topology-Aware MPLS Dataplane Monitoring System", draft-ietf-spring-oam-usecase-03 (work in progress), April 2016.

[LDP-TE] VDE-Verlag, "Traffic Matrices for MPLS Networks with LDP Traffic Statistics", 2004.

Appendix A. ADK2 Test Source Code

The following C++ source code is a modified version of the Code at [RFC6576]. This version allows to test two files containing values with the ADK2. It is not necessary that the values are sorted, because in the first step the values get sorted.

```
/*
Copyright (c) 2012 IETF Trust and the persons identified
as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with
or without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents (http://trustee.ietf.org/license-info).
*/

/* Routines for computing the Anderson-Darling 2 sample
* test statistic.
*
* Implemented based on the description in
* "Anderson-Darling K Sample Test" Heckert, Alan and
* Filliben, James, editors, Dataplot Reference Manual,
* Chapter 15 Auxiliary, NIST, 2004.
* Official Reference by 2010
* Heckert, N. A. (2001). Dataplot website at the
* National Institute of Standards and Technology:
* http://www.itl.nist.gov/div898/software/dataplot.html/
* June 2001.
*/

// this code is a modified version of the code in RFC6576
```

```
// use '-std=c++11' for compiling

#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <iterator>

#include <algorithm>


using namespace std;

/* This function reads the values and sorts this in an ascending
* order.
* The format is: one value per line followed by a line break.
* A blank line at the end of the file will crash the program.
*/
vector<double> read_file_sort (string filename) {
    vector<double> vec;
    // variable for one line of the file and the value
    string line;
    double tmp;

    ifstream file;
    file.open(filename, ios::in);
    if (!file) {
        cout << "Error in file " << filename << endl;
    }
    else {
        // read file in a vector
        while(!file.eof()) {
            getline (file, line);
            tmp = stod (line);
            vec.push_back(tmp);
        }
        // sort the vector ascending
        sort(vec.begin(), vec.end());
    }
    file.close();
    return vec;
}

 int main(int argn, char *argv[]) {

    if (argn != 1 && argn != 3) {
        cout << "wrong invocation" << endl;
        cout << "start with " << argv[0] << " file1 file2" << endl;
```

```
            cout << "start with " << argv[0] << " without parameter, if \
            the files are named file1.csv and file2.csv" << endl;
            return 1;
        }

        vector<double> vec1, vec2;
        double adk_result;
        static int k, val_st_z_samp1, val_st_z_samp2,
                   val_eq_z_samp1, val_eq_z_samp2,
                   j, n_total, n_sample1, n_sample2, L,
                   max_number_samples, line, maxnumber_z;
        static int column_1, column_2;
        static double adk, n_value, z, sum_adk_samp1,
                      sum_adk_samp2, z_aux;
        static double H_j, F1j, hj, F2j, denom_1_aux, denom_2_aux;
        static bool next_z_sample2, equal_z_both_samples;
        static int stop_loop1, stop_loop2, stop_loop3,old_eq_line2,
                   old_eq_line1;

        static double adk_criterium = 1.993;

        string filename1 = "file1.csv";
        string filename2 = "file2.csv";

        // if called with filenames
        if (argn == 3) {
            filename1 = argv[1];
            filename2 = argv[2];
        }

        // sort the two files i a vector
        vec1 = read_file_sort(filename1);
        vec2 = read_file_sort(filename2);

        k = 2;
        n_sample1 = vec1.size() - 1;
        n_sample2 = vec2.size() - 1;

        // -1 because vec[0] is a dummy value
        n_total = n_sample1 + n_sample2;

        /* value equal to the line with a value = zj in sample 1.
         * Here j=1, so the line is 1.
         */
        val_eq_z_samp1 = 1;

        /* value equal to the line with a value = zj in sample 2.
         * Here j=1, so the line is 1.
```

```
        */
       val_eq_z_samp2 = 1;

       /* value equal to the last line with a value < zj
        * in sample 1.  Here j=1, so the line is 0.
        */
       val_st_z_samp1 = 0;

       /* value equal to the last line with a value < zj
        * in sample 1.  Here j=1, so the line is 0.
        */
       val_st_z_samp2 = 0;

       sum_adk_samp1 = 0;
       sum_adk_samp2 = 0;
       j = 1;

       // as mentioned above, j=1
       equal_z_both_samples = false;

       next_z_sample2 = false;

       // assuming the next z to be of sample 1
       stop_loop1 = n_sample1 + 1;

       // + 1 because vec[0] is a dummy, see n_sample1 declaration
       stop_loop2 = n_sample2 + 1;
       stop_loop3 = n_total + 1;

       /* The required z values are calculated until all values
        * of both samples have been taken into account.  See the
        * lines above for the stoploop values.  Construct required
        * to avoid a mathematical operation in the while condition.
       */
       while (((stop_loop1 > val_eq_z_samp1)

              || (stop_loop2 > val_eq_z_samp2)) && stop_loop3 > j) {
           if (val_eq_z_samp1 < n_sample1+1) {
               /* here, a preliminary zj value is set.
                * See below how to calculate the actual zj.
                */
               z = vec1[val_eq_z_samp1];

               /* this while sequence calculates the number of values
                * equal to z.
                */
               while ((val_eq_z_samp1+1 < n_sample1)
                   && z == vec1[val_eq_z_samp1+1] ) {
```

```
                val_eq_z_samp1++;
                }
        }
        else {
        val_eq_z_samp1 = 0;
        val_st_z_samp1 = n_sample1;

        // this should be val_eq_z_samp1 - 1 = n_sample1
        }

        if (val_eq_z_samp2 < n_sample2+1) {
            z_aux = vec2[val_eq_z_samp2];

            /* this while sequence calculates the number of values
             * equal to z_aux
             */

            while ((val_eq_z_samp2+1 < n_sample2)
                    && z_aux == vec2[val_eq_z_samp2+1] ) {
                val_eq_z_samp2++;
            }

            /* the smaller of the two actual data values is picked
             * as the next zj.
             */

            if(z > z_aux) {
                z = z_aux;
                next_z_sample2 = true;
            }
            else {
                if (z == z_aux) {
                equal_z_both_samples = true;
                }

                /* This is the case if the last value of column1 is
                 * smaller than the remaining values of column2.
                 */
               if (val_eq_z_samp1 == 0) {
                z = z_aux;
                next_z_sample2 = true;
                }
            }
        }
        else {
            val_eq_z_samp2 = 0;
            val_st_z_samp2 = n_sample2;
```

```
            // this should be val_eq_z_samp2 - 1 = n_sample2
          }

           /* in the following, sum j = 1 to L is calculated for
            * sample 1 and sample 2.
            */
          if (equal_z_both_samples) {

              /* hj is the number of values in the combined sample
               * equal to zj
               */
              hj = val_eq_z_samp1 - val_st_z_samp1
                + val_eq_z_samp2 - val_st_z_samp2;

              /* H_j is the number of values in the combined sample
               * smaller than zj plus one half the number of
               * values in the combined sample equal to zj
               * (that's hj/2).
               */
              H_j = val_st_z_samp1 + val_st_z_samp2 + hj / 2;

              /* F1j is the number of values in the 1st sample
               * that are less than zj plus one half the number
               * of values in this sample that are equal to zj.
               */

              F1j = val_st_z_samp1 + (double)
                    (val_eq_z_samp1 - val_st_z_samp1) / 2;

              /* F2j is the number of values in the 1st sample
               * that are less than zj plus one half the number
               * of values in this sample that are equal to zj.
               */
              F2j = val_st_z_samp2 + (double)
                   (val_eq_z_samp2 - val_st_z_samp2) / 2;

              /* set the line of values equal to zj to the
               * actual line of the last value picked for zj.
               */
              val_st_z_samp1 = val_eq_z_samp1;

              /* Set the line of values equal to zj to the actual
               * line of the last value picked for zj of each
               * sample.  This is required as data smaller than zj
               * is accounted differently than values equal to zj.
               */
              val_st_z_samp2 = val_eq_z_samp2;
```

```
            /* next the lines of the next values z, i.e., zj+1
             * are addressed.
             */
            val_eq_z_samp1++;

            /* next the lines of the next values z, i.e.,
             * zj+1 are addressed
             */
            val_eq_z_samp2++;
        }
        else {

            /* the smaller z value was contained in sample 2;
             * hence, this value is the zj to base the following
             * calculations on.
             */
            if (next_z_sample2){
                /* hj is the number of values in the combined
                 * sample equal to zj; in this case, these are
                 * within sample 2 only.
                 */
                hj = val_eq_z_samp2 - val_st_z_samp2;

                /* H_j is the number of values in the combined sample
                 * smaller than zj plus one half the number of
                 * values in the combined sample equal to zj
                 * (that's hj/2).
                 */
                H_j = val_st_z_samp1 + val_st_z_samp2 + hj / 2;

              /* F1j is the number of values in the 1st sample that
             * are less than zj plus one half the number of values in
               * this sample that are equal to zj.
               * As val_eq_z_samp2 < val_eq_z_samp1, these are the
               * val_st_z_samp1 only.
               */
                F1j = val_st_z_samp1;

              /* F2j is the number of values in the 1st sample that
             * are less than zj plus one half the number of values in
             * this sample that are equal to zj.  The latter are from
               * sample 2 only in this case.
               */

                F2j = val_st_z_samp2 + (double)
                     (val_eq_z_samp2 - val_st_z_samp2) / 2;

             /* Set the line of values equal to zj to the actual line
```

```
                * of the last value picked for zj of sample 2 only in
                * this case.
                */
                val_st_z_samp2 = val_eq_z_samp2;

               /* next the line of the next value z, i.e., zj+1 is
                * addressed.  Here, only sample 2 must be addressed.
                */

                val_eq_z_samp2++;
                if (val_eq_z_samp1 == 0) {
                    val_eq_z_samp1 = stop_loop1;
                }
            }
           /* the smaller z value was contained in sample 2;
            * hence, this value is the zj to base the following
            * calculations on.
            */

            else {

                /* hj is the number of values in the combined
                 * sample equal to zj; in this case, these are
                 * within sample 1 only.
                 */
                hj = val_eq_z_samp1 - val_st_z_samp1;

                /* H_j is the number of values in the combined
                 * sample smaller than zj plus one half the number
                 * of values in the combined sample equal to zj
                 * (that's hj/2).
                 */

                H_j = val_st_z_samp1 + val_st_z_samp2 + hj / 2;

            /* F1j is the number of values in the 1st sample that
             * are less than zj plus; in this case, these are within
             * sample 1 only one half the number of values in this
             * sample that are equal to zj.  The latter are from
             * sample 1 only in this case.
             */

                F1j = val_st_z_samp1 + (double)
                    (val_eq_z_samp1 - val_st_z_samp1) / 2;

            /* F2j is the number of values in the 1st sample that
             * are less than zj plus one half the number of values
             * in this sample that are equal to zj.  As
```

```
                * val_eq_z_samp1 < val_eq_z_samp2, these are the
                * val_st_z_samp2 only.
                */

                   F2j = val_st_z_samp2;

               /* Set the line of values equal to zj to the actual line
                * of the last value picked for zj of sample 1 only in
                * this case.
                */

                   val_st_z_samp1 = val_eq_z_samp1;
                   /* next the line of the next value z, i.e., zj+1 is
                    * addressed.  Here, only sample 1 must be addressed.
                    */
                   val_eq_z_samp1++;

                   if (val_eq_z_samp2 == 0) {
                       val_eq_z_samp2 = stop_loop2;
                   }
               }
           }

           denom_1_aux = n_total * F1j - n_sample1 * H_j;
           denom_2_aux = n_total * F2j - n_sample2 * H_j;

           sum_adk_samp1 = sum_adk_samp1 + hj
                           * (denom_1_aux * denom_1_aux) /
                           (H_j * (n_total - H_j)
                           - n_total * hj / 4);
           sum_adk_samp2 = sum_adk_samp2 + hj
                           * (denom_2_aux * denom_2_aux) /
                           (H_j * (n_total - H_j)
                           - n_total * hj / 4);

           next_z_sample2 = false;
           equal_z_both_samples = false;

           /* index to count the z.  It is only required to prevent
            * the while slope to execute endless
            */
           j++;
       }

       // calculating the adk value is the final step.
       adk_result = (double) (n_total - 1) / (n_total
              * n_total * (k - 1))
               * (sum_adk_samp1 / n_sample1
```

```
                + sum_adk_samp2 / n_sample2);


        /* if(adk_result <= adk_criterium)
         * adk_2_sample test is passed
         */
        //return adk_result <= adk_criterium;
        cout << "Result: " << adk_result << endl;
    }
```

Authors' Addresses

Raik Leipnitz (editor)
Deutsche Telekom
Olgastr. 67
Ulm 89073
Germany

Email: r.leipnitz@telekom.de

Ruediger Geib
Deutsche Telekom
Heinrich Hertz Str. 3-7
Darmstadt 64295
Germany

Phone: +49 6151 5812747
Email: Ruediger.Geib@telekom.de

PCE Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

D. Dhody, Ed.
Huawei Technologies
J. Hardwick
Metaswitch
V. Beeram
Juniper Networks
J. Tantsura
July 7, 2016

A YANG Data Model for Path Computation Element Communications Protocol (PCEP)
draft-pkd-pce-pcep-yang-06

Abstract

This document defines a YANG data model for the management of Path Computation Element communications Protocol (PCEP) for communications between a Path Computation Client (PCC) and a Path Computation Element (PCE), or between two PCEs. The data model includes configuration data and state data (status information and counters for the collection of statistics).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Requirements Language 3
3. Terminology and Notation 3
3.1. Tree Diagrams 4
3.2. Prefixes in Data Node Names 5
4. Objectives 5
5. The Design of PCEP Data Model 6
5.1. The Entity 17
5.2. The Peer Lists 18
5.3. The Session Lists 18
5.4. Notifications 19
6. Advanced PCE Features 19
6.1. Stateful PCE's LSP-DB 19
7. Open Issues and Next Step 20
7.1. The PCE-Initiated LSP 20
7.2. PCEP over TLS (PCEPS) 20
8. PCEP YANG Module 20
9. Security Considerations 83
10. Manageability Considerations 84
10.1. Control of Function and Policy 84
10.2. Information and Data Models 84
10.3. Liveness Detection and Monitoring 84
10.4. Verify Correct Operations 84
10.5. Requirements On Other Protocols 84
10.6. Impact On Network Operations 84
11. IANA Considerations 84
12. Acknowledgements 85
13. References 85
13.1. Normative References 85
13.2. Informative References 86
Appendix A. Contributor Addresses 88
Authors' Addresses 89

1. Introduction

The Path Computation Element (PCE) defined in [RFC4655] is an entity that is capable of computing a network path or route based on a network graph, and applying computational constraints. A Path

Computation Client (PCC) may make requests to a PCE for paths to be computed.

PCEP is the communication protocol between a PCC and PCE and is defined in [RFC5440]. PCEP interactions include path computation requests and path computation replies as well as notifications of specific states related to the use of a PCE in the context of Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) Traffic Engineering (TE). [I-D.ietf-pce-stateful-pce] specifies extensions to PCEP to enable stateful control of MPLS TE LSPs.

This document defines a YANG [RFC6020] data model for the management of PCEP speakers. It is important to establish a common data model for how PCEP speakers are identified, configured, and monitored. The data model includes configuration data and state data (status information and counters for the collection of statistics).

This document contains a specification of the PCEP YANG module, "ietf-pcep" which provides the PCEP [RFC5440] data model.

2. 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].

3. Terminology and Notation

This document uses the terminology defined in [RFC4655] and [RFC5440]. In particular, it uses the following acronyms.

- Path Computation Request message (PCReq).
- Path Computation Reply message (PCRep).
- Notification message (PCNtf).
- Error message (PCErr).
- Request Parameters object (RP).
- Synchronization Vector object (SVEC).
- Explicit Route object (ERO).

This document also uses the following terms defined in [RFC7420]:

- PCEP entity: a local PCEP speaker.

- PCEP peer: to refer to a remote PCEP speaker.
- PCEP speaker: where it is not necessary to distinguish between local and remote.

Further, this document also uses the following terms defined in [I-D.ietf-pce-stateful-pce] :

- Stateful PCE, Passive Stateful PCE, Active Stateful PCE
- Delegation, Revocation, Redelegation
- LSP State Report, Path Computation Report message (PCRpt).
- LSP State Update, Path Computation Update message (PCUpd).

[I-D.ietf-pce-pce-initiated-lsp] :

- PCE-initiated LSP, Path Computation LSP Initiate Message (PCInitiate).

[I-D.ietf-pce-lsp-setup-type] :

- Path Setup Type (PST).

[I-D.ietf-pce-segment-routing] :

- Segment Routing (SR).
- Segment Identifier (SID).
- Maximum SID Depth (MSD).

3.1. Tree Diagrams

A graphical representation of the complete data tree is presented in Section 5. The meaning of the symbols in these diagrams is as follows and as per [I-D.ietf-netmod-rfc6087bis]:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

3.2. 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

4. Objectives

This section describes 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.
- Mapping to the PCEP MIB Module should be clear.
- The data model should allow for static configurations of peers.
- The data model should include read-only counters in order to gather statistics for sent and received PCEP messages, received messages with errors, and messages that could not be sent due to errors.

   o It should be fairly straightforward to augment the base data model for advanced PCE features.

5. The Design of PCEP Data Model

The module, "ietf-pcep", defines the basic components of a PCE speaker.

```
module: ietf-pcep
   +--rw pcep!
   |  +--rw entity
   |     +--rw addr                          inet:ip-address
   |     +--rw enabled?                      boolean
   |     +--rw role                          pcep-role
   |     +--rw description?                  string
   |     +--rw domain
   |     |  +--rw domain* [domain-type domain]
   |     |     +--rw domain-type    domain-type
   |     |     +--rw domain         domain
   |     +--rw capability
   |     |  +--rw gmpls?                 boolean {gmpls}?
   |     |  +--rw bi-dir?                boolean
   |     |  +--rw diverse?               boolean
   |     |  +--rw load-balance?          boolean
   |     |  +--rw synchronize?           boolean {svec}?
   |     |  +--rw objective-function?    boolean {obj-fn}?
   |     |  +--rw add-path-constraint?   boolean
   |     |  +--rw prioritization?        boolean
   |     |  +--rw multi-request?         boolean
   |     |  +--rw gco?                   boolean {gco}?
   |     |  +--rw p2mp?                  boolean {p2mp}?
   |     |  +--rw stateful {stateful}?
   |     |  |  +--rw enabled?         boolean
   |     |  |  +--rw active?          boolean
   |     |  |  +--rw pce-initiated?   boolean {pce-initiated}?
   |     |  +--rw sr {sr}?
   |     |     +--rw enabled?   boolean
   |     |     +--rw msd?       uint8
   |     +--rw pce-info
   |     |  +--rw scope
   |     |  |  +--rw intra-area-scope?           boolean
   |     |  |  +--rw intra-area-pref?            uint8
   |     |  |  +--rw inter-area-scope?           boolean
   |     |  |  +--rw inter-area-scope-default?   boolean
   |     |  |  +--rw inter-area-pref?            uint8
   |     |  |  +--rw inter-as-scope?             boolean
   |     |  |  +--rw inter-as-scope-default?     boolean
   |     |  |  +--rw inter-as-pref?              uint8
```

```
|     |  |  +--rw inter-layer-scope?          boolean
|     |  |  +--rw inter-layer-pref?           uint8
|     |  +--rw neigh-domains
|     |  |  +--rw domain* [domain-type domain]
|     |  |     +--rw domain-type    domain-type
|     |  |     +--rw domain         domain
|     |  +--rw (auth-type-selection)?
|     |     +--:(auth-key-chain)
|     |     |  +--rw key-chain?          key-chain:key-chain-ref
|     |     +--:(auth-key)
|     |     |  +--rw key?                string
|     |     |  +--rw crypto-algorithm
|     |     |     +--rw (algorithm)?
|     |     |        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
|     |     |        |  +--rw hmac-sha1-12?             empty
|     |     |        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
|     |     |        |  +--rw aes-cmac-prf-128?         empty
|     |     |        +--:(md5)
|     |     |        |  +--rw md5?                      empty
|     |     |        +--:(sha-1)
|     |     |        |  +--rw sha-1?                    empty
|     |     |        +--:(hmac-sha-1)
|     |     |        |  +--rw hmac-sha-1?               empty
|     |     |        +--:(hmac-sha-256)
|     |     |        |  +--rw hmac-sha-256?             empty
|     |     |        +--:(hmac-sha-384)
|     |     |        |  +--rw hmac-sha-384?             empty
|     |     |        +--:(hmac-sha-512)
|     |     |        |  +--rw hmac-sha-512?             empty
|     |     |        +--:(clear-text) {clear-text}?
|     |     |        |  +--rw clear-text?               empty
|     |     |        +--:(replay-protection-only) {replay-protection-only}?
|     |     |           +--rw replay-protection-only?   empty
|     |     +--:(auth-tls) {tls}?
|     |        +--rw tls
|     +--rw connect-timer?               uint32
|     +--rw connect-max-retry?           uint32
|     +--rw init-backoff-timer?          uint32
|     +--rw max-backoff-timer?           uint32
|     +--rw open-wait-timer?             uint32
|     +--rw keep-wait-timer?             uint32
|     +--rw keep-alive-timer?            uint32
|     +--rw dead-timer?                  uint32
|     +--rw allow-negotiation?           boolean
|     +--rw max-keep-alive-timer?        uint32
|     +--rw max-dead-timer?              uint32
|     +--rw min-keep-alive-timer?        uint32
|     +--rw min-dead-timer?              uint32
```

```
|     +--rw sync-timer?                   uint32 {svec}?
|     +--rw request-timer?                uint32
|     +--rw max-sessions?                 uint32
|     +--rw max-unknown-reqs?             uint32
|     +--rw max-unknown-msgs?             uint32
|     +--rw pcep-notification-max-rate    uint32
|     +--rw stateful-parameter {stateful}?
|     |  +--rw state-timeout?          uint32
|     |  +--rw redelegation-timeout?   uint32
|     |  +--rw rpt-non-pcep-lsp?       boolean
|     +--rw peers
|        +--rw peer* [addr]
|           +--rw addr                inet:ip-address
|           +--rw description?        string
|           +--rw domain
|           |  +--rw domain* [domain-type domain]
|           |     +--rw domain-type    domain-type
|           |     +--rw domain         domain
|           +--rw capability
|           |  +--rw gmpls?                 boolean {gmpls}?
|           |  +--rw bi-dir?                boolean
|           |  +--rw diverse?               boolean
|           |  +--rw load-balance?          boolean
|           |  +--rw synchronize?           boolean {svec}?
|           |  +--rw objective-function?    boolean {obj-fn}?
|           |  +--rw add-path-constraint?   boolean
|           |  +--rw prioritization?        boolean
|           |  +--rw multi-request?         boolean
|           |  +--rw gco?                   boolean {gco}?
|           |  +--rw p2mp?                  boolean {p2mp}?
|           |  +--rw stateful {stateful}?
|           |  |  +--rw enabled?         boolean
|           |  |  +--rw active?          boolean
|           |  |  +--rw pce-initiated?   boolean {pce-initiated}?
|           |  +--rw sr {sr}?
|           |     +--rw enabled?   boolean
|           |     +--rw msd?       uint8
|           +--rw scope
|           |  +--rw intra-area-scope?           boolean
|           |  +--rw intra-area-pref?            uint8
|           |  +--rw inter-area-scope?           boolean
|           |  +--rw inter-area-scope-default?   boolean
|           |  +--rw inter-area-pref?            uint8
|           |  +--rw inter-as-scope?             boolean
|           |  +--rw inter-as-scope-default?     boolean
|           |  +--rw inter-as-pref?              uint8
|           |  +--rw inter-layer-scope?          boolean
|           |  +--rw inter-layer-pref?           uint8
```

```
   |           +--rw neigh-domains
   |           |  +--rw domain* [domain-type domain]
   |           |     +--rw domain-type    domain-type
   |           |     +--rw domain         domain
   |           +--rw delegation-pref?    uint8 {stateful}?
   |           +--rw (auth-type-selection)?
   |              +--:(auth-key-chain)
   |              |  +--rw key-chain?          key-chain:key-chain-ref
   |              +--:(auth-key)
   |              |  +--rw key?                string
   |              |  +--rw crypto-algorithm
   |              |     +--rw (algorithm)?
   |              |        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
   |              |        |  +--rw hmac-sha1-12?             empty
   |              |        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
   |              |        |  +--rw aes-cmac-prf-128?         empty
   |              |        +--:(md5)
   |              |        |  +--rw md5?                      empty
   |              |        +--:(sha-1)
   |              |        |  +--rw sha-1?                    empty
   |              |        +--:(hmac-sha-1)
   |              |        |  +--rw hmac-sha-1?               empty
   |              |        +--:(hmac-sha-256)
   |              |        |  +--rw hmac-sha-256?             empty
   |              |        +--:(hmac-sha-384)
   |              |        |  +--rw hmac-sha-384?             empty
   |              |        +--:(hmac-sha-512)
   |              |        |  +--rw hmac-sha-512?             empty
   |              |        +--:(clear-text) {clear-text}?
   |              |        |  +--rw clear-text?               empty
   |              |        +--:(replay-protection-only) {replay-protection-only}
?
   |              |           +--rw replay-protection-only?   empty
   |              +--:(auth-tls) {tls}?
   |                 +--rw tls
   +--ro pcep-state
      +--ro entity
         +--ro addr?                  inet:ip-address
         +--ro index?                 uint32
         +--ro admin-status?          pcep-admin-status
         +--ro oper-status?           pcep-admin-status
         +--ro role?                  pcep-role
         +--ro domain
         |  +--ro domain* [domain-type domain]
         |     +--ro domain-type    domain-type
         |     +--ro domain         domain
         +--ro capability
         |  +--ro gmpls?                boolean {gmpls}?
         |  +--ro bi-dir?               boolean
```

```
|  +--ro diverse?               boolean
|  +--ro load-balance?          boolean
|  +--ro synchronize?           boolean {svec}?
|  +--ro objective-function?    boolean {obj-fn}?
|  +--ro add-path-constraint?   boolean
|  +--ro prioritization?        boolean
|  +--ro multi-request?         boolean
|  +--ro gco?                   boolean {gco}?
|  +--ro p2mp?                  boolean {p2mp}?
|  +--ro stateful {stateful}?
|  |  +--ro enabled?         boolean
|  |  +--ro active?          boolean
|  |  +--ro pce-initiated?   boolean {pce-initiated}?
|  +--ro sr {sr}?
|     +--ro enabled?   boolean
|     +--ro msd?       uint8
+--ro pce-info
|  +--ro scope
|  |  +--ro intra-area-scope?           boolean
|  |  +--ro intra-area-pref?            uint8
|  |  +--ro inter-area-scope?           boolean
|  |  +--ro inter-area-scope-default?   boolean
|  |  +--ro inter-area-pref?            uint8
|  |  +--ro inter-as-scope?             boolean
|  |  +--ro inter-as-scope-default?     boolean
|  |  +--ro inter-as-pref?              uint8
|  |  +--ro inter-layer-scope?          boolean
|  |  +--ro inter-layer-pref?           uint8
|  +--ro neigh-domains
|  |  +--ro domain* [domain-type domain]
|  |     +--ro domain-type    domain-type
|  |     +--ro domain         domain
|  +--ro (auth-type-selection)?
|     +--:(auth-key-chain)
|     |  +--ro key-chain?          key-chain:key-chain-ref
|     +--:(auth-key)
|     |  +--ro key?                string
|     |  +--ro crypto-algorithm
|     |     +--ro (algorithm)?
|     |        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
|     |        |  +--ro hmac-sha1-12?             empty
|     |        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
|     |        |  +--ro aes-cmac-prf-128?         empty
|     |        +--:(md5)
|     |        |  +--ro md5?                      empty
|     |        +--:(sha-1)
|     |        |  +--ro sha-1?                    empty
|     |        +--:(hmac-sha-1)
```

```
|     |       |  +--ro hmac-sha-1?              empty
|     |       +--:(hmac-sha-256)
|     |       |  +--ro hmac-sha-256?            empty
|     |       +--:(hmac-sha-384)
|     |       |  +--ro hmac-sha-384?            empty
|     |       +--:(hmac-sha-512)
|     |       |  +--ro hmac-sha-512?            empty
|     |       +--:(clear-text) {clear-text}?
|     |       |  +--ro clear-text?              empty
|     |       +--:(replay-protection-only) {replay-protection-only}?
|     |          +--ro replay-protection-only?  empty
|     +--:(auth-tls) {tls}?
|        +--ro tls
+--ro connect-timer?          uint32
+--ro connect-max-retry?      uint32
+--ro init-backoff-timer?     uint32
+--ro max-backoff-timer?      uint32
+--ro open-wait-timer?        uint32
+--ro keep-wait-timer?        uint32
+--ro keep-alive-timer?       uint32
+--ro dead-timer?             uint32
+--ro allow-negotiation?      boolean
+--ro max-keep-alive-timer?   uint32
+--ro max-dead-timer?         uint32
+--ro min-keep-alive-timer?   uint32
+--ro min-dead-timer?         uint32
+--ro sync-timer?             uint32 {svec}?
+--ro request-timer?          uint32
+--ro max-sessions?           uint32
+--ro max-unknown-reqs?       uint32
+--ro max-unknown-msgs?       uint32
+--ro stateful-parameter {stateful}?
|  +--ro state-timeout?         uint32
|  +--ro redelegation-timeout?  uint32
|  +--ro rpt-non-pcep-lsp?      boolean
+--ro lsp-db {stateful}?
|  +--ro association-list* [id source global-source extended-id]
|  |  +--ro type?            assoc-type
|  |  +--ro id               uint16
|  |  +--ro source           inet:ip-address
|  |  +--ro global-source    uint32
|  |  +--ro extended-id      string
|  |  +--ro lsp* [plsp-id pcc-id]
|  |     +--ro plsp-id    -> /pcep-state/entity/lsp-db/lsp/plsp-id
|  |     +--ro pcc-id     -> /pcep-state/entity/lsp-db/lsp/pcc-id
|  +--ro lsp* [plsp-id pcc-id]
|     +--ro plsp-id              uint32
|     +--ro pcc-id               inet:ip-address
```

```
         |     +--ro lsp-ref
         |     |  +--ro source?               -> /te:te/lsps-state/lsp/source
         |     |  +--ro destination?          -> /te:te/lsps-state/lsp/destinati
on
         |     |  +--ro tunnel-id?            -> /te:te/lsps-state/lsp/tunnel-id
         |     |  +--ro lsp-id?               -> /te:te/lsps-state/lsp/lsp-id
         |     |  +--ro extended-tunnel-id?   -> /te:te/lsps-state/lsp/extended-
tunnel-id
         |     |  +--ro type?                 -> /te:te/lsps-state/lsp/type
         |     +--ro admin-state?          boolean
         |     +--ro operational-state?    operational-state
         |     +--ro delegated
         |     |  +--ro enabled?   boolean
         |     |  +--ro pce?       -> /pcep-state/entity/peers/peer/addr
         |     |  +--ro srp-id?    uint32
         |     +--ro initiation {pce-initiated}?
         |     |  +--ro enabled?   boolean
         |     |  +--ro pce?       -> /pcep-state/entity/peers/peer/addr
         |     +--ro symbolic-path-name?   string
         |     +--ro last-error?           lsp-error
         |     +--ro pst?                  pst
         |     +--ro association-list* [id source global-source extended-id]
         |        +--ro id               -> /pcep-state/entity/lsp-db/associatio
n-list/id
         |        +--ro source           -> /pcep-state/entity/lsp-db/associatio
n-list/source
         |        +--ro global-source    -> /pcep-state/entity/lsp-db/associatio
n-list/global-source
         |        +--ro extended-id      -> /pcep-state/entity/lsp-db/associatio
n-list/extended-id
         +--ro peers
            +--ro peer* [addr]
               +--ro addr                    inet:ip-address
               +--ro role?                   pcep-role
               +--ro domain
               |  +--ro domain* [domain-type domain]
               |     +--ro domain-type    domain-type
               |     +--ro domain         domain
               +--ro capability
               |  +--ro gmpls?                boolean {gmpls}?
               |  +--ro bi-dir?               boolean
               |  +--ro diverse?              boolean
               |  +--ro load-balance?         boolean
               |  +--ro synchronize?          boolean {svec}?
               |  +--ro objective-function?   boolean {obj-fn}?
               |  +--ro add-path-constraint?  boolean
               |  +--ro prioritization?       boolean
               |  +--ro multi-request?        boolean
               |  +--ro gco?                  boolean {gco}?
               |  +--ro p2mp?                 boolean {p2mp}?
               |  +--ro stateful {stateful}?
               |  |  +--ro enabled?         boolean
               |  |  +--ro active?          boolean
               |  |  +--ro pce-initiated?   boolean {pce-initiated}?
```

```
               |  +--ro sr {sr}?
               |     +--ro enabled?   boolean
               |     +--ro msd?       uint8
               +--ro pce-info
               |  +--ro scope
               |  |  +--ro intra-area-scope?           boolean
               |  |  +--ro intra-area-pref?            uint8
               |  |  +--ro inter-area-scope?           boolean
               |  |  +--ro inter-area-scope-default?   boolean
               |  |  +--ro inter-area-pref?            uint8
               |  |  +--ro inter-as-scope?             boolean
               |  |  +--ro inter-as-scope-default?     boolean
               |  |  +--ro inter-as-pref?              uint8
               |  |  +--ro inter-layer-scope?          boolean
               |  |  +--ro inter-layer-pref?           uint8
               |  +--ro neigh-domains
               |     +--ro domain* [domain-type domain]
               |        +--ro domain-type    domain-type
               |        +--ro domain         domain
               +--ro delegation-pref?        uint8 {stateful}?
               +--ro (auth-type-selection)?
               |  +--:(auth-key-chain)
               |  |  +--ro key-chain?              key-chain:key-chain-ref
               |  +--:(auth-key)
               |  |  +--ro key?                    string
               |  |  +--ro crypto-algorithm
               |  |     +--ro (algorithm)?
               |  |        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
               |  |        |  +--ro hmac-sha1-12?             empty
               |  |        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
               |  |        |  +--ro aes-cmac-prf-128?         empty
               |  |        +--:(md5)
               |  |        |  +--ro md5?                     empty
               |  |        +--:(sha-1)
               |  |        |  +--ro sha-1?                   empty
               |  |        +--:(hmac-sha-1)
               |  |        |  +--ro hmac-sha-1?              empty
               |  |        +--:(hmac-sha-256)
               |  |        |  +--ro hmac-sha-256?            empty
               |  |        +--:(hmac-sha-384)
               |  |        |  +--ro hmac-sha-384?            empty
               |  |        +--:(hmac-sha-512)
               |  |        |  +--ro hmac-sha-512?            empty
               |  |        +--:(clear-text) {clear-text}?
               |  |        |  +--ro clear-text?              empty
               |  |        +--:(replay-protection-only) {replay-protection-only}
?
               |  |           +--ro replay-protection-only?   empty
               |  +--:(auth-tls) {tls}?
```

```
|     +--ro tls
+--ro discontinuity-time?     yang:timestamp
+--ro initiate-session?       boolean
+--ro session-exists?         boolean
+--ro num-sess-setup-ok?      yang:counter32
+--ro num-sess-setup-fail?    yang:counter32
+--ro session-up-time?        yang:timestamp
+--ro session-fail-time?      yang:timestamp
+--ro session-fail-up-time?   yang:timestamp
+--ro pcep-stats
|  +--ro avg-rsp-time?               uint32
|  +--ro lwm-rsp-time?               uint32
|  +--ro hwm-rsp-time?               uint32
|  +--ro num-pcreq-sent?             yang:counter32
|  +--ro num-pcreq-rcvd?             yang:counter32
|  +--ro num-pcrep-sent?             yang:counter32
|  +--ro num-pcrep-rcvd?             yang:counter32
|  +--ro num-pcerr-sent?             yang:counter32
|  +--ro num-pcerr-rcvd?             yang:counter32
|  +--ro num-pcntf-sent?             yang:counter32
|  +--ro num-pcntf-rcvd?             yang:counter32
|  +--ro num-keepalive-sent?         yang:counter32
|  +--ro num-keepalive-rcvd?         yang:counter32
|  +--ro num-unknown-rcvd?           yang:counter32
|  +--ro num-corrupt-rcvd?           yang:counter32
|  +--ro num-req-sent?               yang:counter32
|  +--ro num-req-sent-pend-rep?      yang:counter32
|  +--ro num-req-sent-ero-rcvd?      yang:counter32
|  +--ro num-req-sent-nopath-rcvd?   yang:counter32
|  +--ro num-req-sent-cancel-rcvd?   yang:counter32
|  +--ro num-req-sent-error-rcvd?    yang:counter32
|  +--ro num-req-sent-timeout?       yang:counter32
|  +--ro num-req-sent-cancel-sent?   yang:counter32
|  +--ro num-req-rcvd?               yang:counter32
|  +--ro num-req-rcvd-pend-rep?      yang:counter32
|  +--ro num-req-rcvd-ero-sent?      yang:counter32
|  +--ro num-req-rcvd-nopath-sent?   yang:counter32
|  +--ro num-req-rcvd-cancel-sent?   yang:counter32
|  +--ro num-req-rcvd-error-sent?    yang:counter32
|  +--ro num-req-rcvd-cancel-rcvd?   yang:counter32
|  +--ro num-rep-rcvd-unknown?       yang:counter32
|  +--ro num-req-rcvd-unknown?       yang:counter32
|  +--ro svec {svec}?
|  |  +--ro num-svec-sent?       yang:counter32
|  |  +--ro num-svec-req-sent?   yang:counter32
|  |  +--ro num-svec-rcvd?       yang:counter32
|  |  +--ro num-svec-req-rcvd?   yang:counter32
|  +--ro stateful {stateful}?
```

```
|  |  +--ro num-pcrpt-sent?             yang:counter32
|  |  +--ro num-pcrpt-rcvd?             yang:counter32
|  |  +--ro num-pcupd-sent?             yang:counter32
|  |  +--ro num-pcupd-rcvd?             yang:counter32
|  |  +--ro num-rpt-sent?               yang:counter32
|  |  +--ro num-rpt-rcvd?               yang:counter32
|  |  +--ro num-rpt-rcvd-error-sent?    yang:counter32
|  |  +--ro num-upd-sent?               yang:counter32
|  |  +--ro num-upd-rcvd?               yang:counter32
|  |  +--ro num-upd-rcvd-unknown?       yang:counter32
|  |  +--ro num-upd-rcvd-undelegated?   yang:counter32
|  |  +--ro num-upd-rcvd-error-sent?    yang:counter32
|  |  +--ro initiation {pce-initiated}?
|  |     +--ro num-pcinitiate-sent?            yang:counter32
|  |     +--ro num-pcinitiate-rcvd?            yang:counter32
|  |     +--ro num-initiate-sent?              yang:counter32
|  |     +--ro num-initiate-rcvd?              yang:counter32
|  |     +--ro num-initiate-rcvd-error-sent?   yang:counter32
|  +--ro num-req-sent-closed?        yang:counter32
|  +--ro num-req-rcvd-closed?        yang:counter32
+--ro sessions
   +--ro session* [initiator]
      +--ro initiator               pcep-initiator
      +--ro state-last-change?      yang:timestamp
      +--ro state?                  pcep-sess-state
      +--ro session-creation?       yang:timestamp
      +--ro connect-retry?          yang:counter32
      +--ro local-id?               uint32
      +--ro remote-id?              uint32
      +--ro keepalive-timer?        uint32
      +--ro peer-keepalive-timer?   uint32
      +--ro dead-timer?             uint32
      +--ro peer-dead-timer?        uint32
      +--ro ka-hold-time-rem?       uint32
      +--ro overloaded?             boolean
      +--ro overload-time?          uint32
      +--ro peer-overloaded?        boolean
      +--ro peer-overload-time?     uint32
      +--ro lspdb-sync?             sync-state {stateful}?
      +--ro discontinuity-time?     yang:timestamp
      +--ro pcep-stats
         +--ro avg-rsp-time?              uint32
         +--ro lwm-rsp-time?              uint32
         +--ro hwm-rsp-time?              uint32
         +--ro num-pcreq-sent?            yang:counter32
         +--ro num-pcreq-rcvd?            yang:counter32
         +--ro num-pcrep-sent?            yang:counter32
         +--ro num-pcrep-rcvd?            yang:counter32
```

```
                        +--ro num-pcerr-sent?             yang:counter32
                        +--ro num-pcerr-rcvd?             yang:counter32
                        +--ro num-pcntf-sent?             yang:counter32
                        +--ro num-pcntf-rcvd?             yang:counter32
                        +--ro num-keepalive-sent?         yang:counter32
                        +--ro num-keepalive-rcvd?         yang:counter32
                        +--ro num-unknown-rcvd?           yang:counter32
                        +--ro num-corrupt-rcvd?           yang:counter32
                        +--ro num-req-sent?               yang:counter32
                        +--ro num-req-sent-pend-rep?      yang:counter32
                        +--ro num-req-sent-ero-rcvd?      yang:counter32
                        +--ro num-req-sent-nopath-rcvd?   yang:counter32
                        +--ro num-req-sent-cancel-rcvd?   yang:counter32
                        +--ro num-req-sent-error-rcvd?    yang:counter32
                        +--ro num-req-sent-timeout?       yang:counter32
                        +--ro num-req-sent-cancel-sent?   yang:counter32
                        +--ro num-req-rcvd?               yang:counter32
                        +--ro num-req-rcvd-pend-rep?      yang:counter32
                        +--ro num-req-rcvd-ero-sent?      yang:counter32
                        +--ro num-req-rcvd-nopath-sent?   yang:counter32
                        +--ro num-req-rcvd-cancel-sent?   yang:counter32
                        +--ro num-req-rcvd-error-sent?    yang:counter32
                        +--ro num-req-rcvd-cancel-rcvd?   yang:counter32
                        +--ro num-rep-rcvd-unknown?       yang:counter32
                        +--ro num-req-rcvd-unknown?       yang:counter32
                        +--ro svec {svec}?
                        |  +--ro num-svec-sent?       yang:counter32
                        |  +--ro num-svec-req-sent?   yang:counter32
                        |  +--ro num-svec-rcvd?       yang:counter32
                        |  +--ro num-svec-req-rcvd?   yang:counter32
                        +--ro stateful {stateful}?
                           +--ro num-pcrpt-sent?             yang:counter32
                           +--ro num-pcrpt-rcvd?             yang:counter32
                           +--ro num-pcupd-sent?             yang:counter32
                           +--ro num-pcupd-rcvd?             yang:counter32
                           +--ro num-rpt-sent?               yang:counter32
                           +--ro num-rpt-rcvd?               yang:counter32
                           +--ro num-rpt-rcvd-error-sent?    yang:counter32
                           +--ro num-upd-sent?               yang:counter32
                           +--ro num-upd-rcvd?               yang:counter32
                           +--ro num-upd-rcvd-unknown?       yang:counter32
                           +--ro num-upd-rcvd-undelegated?   yang:counter32
                           +--ro num-upd-rcvd-error-sent?    yang:counter32
                           +--ro initiation {pce-initiated}?
                              +--ro num-pcinitiate-sent?           yang:counter
32
                              +--ro num-pcinitiate-rcvd?           yang:counter
32
                              +--ro num-initiate-sent?             yang:counter
32
                              +--ro num-initiate-rcvd?             yang:counter
32
```

```
                              +--ro num-initiate-rcvd-error-sent?   yang:counter
32
notifications:
   +---n pcep-session-up
   |  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   |  +--ro session-initiator?   -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
   |  +--ro state-last-change?   yang:timestamp
   |  +--ro state?               pcep-sess-state
   +---n pcep-session-down
   |  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   |  +--ro session-initiator?   pcep-initiator
   |  +--ro state-last-change?   yang:timestamp
   |  +--ro state?               pcep-sess-state
   +---n pcep-session-local-overload
   |  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   |  +--ro session-initiator?   -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
   |  +--ro overloaded?          boolean
   |  +--ro overload-time?       uint32
   +---n pcep-session-local-overload-clear
   |  +--ro peer-addr?    -> /pcep-state/entity/peers/peer/addr
   |  +--ro overloaded?   boolean
   +---n pcep-session-peer-overload
   |  +--ro peer-addr?            -> /pcep-state/entity/peers/peer/addr
   |  +--ro session-initiator?    -> /pcep-state/entity/peers/peer/sessions/sess
ion/initiator
   |  +--ro peer-overloaded?      boolean
   |  +--ro peer-overload-time?   uint32
   +---n pcep-session-peer-overload-clear
      +--ro peer-addr?         -> /pcep-state/entity/peers/peer/addr
      +--ro peer-overloaded?   boolean
```

5.1. The Entity

The PCEP yang module may contain status information for the local PCEP entity.

The entity has an IP address (using ietf-inet-types [RFC6991]) and a "role" leaf (the local entity PCEP role) as mandatory.

Note that, the PCEP MIB module [RFC7420] uses an entity list and a system generated entity index as a primary index to the read only entity table. If the device implements the PCEP MIB, the "index" leaf MUST contain the value of the corresponding pcePcepEntityIndex and only one entity is assumed.

5.2. The Peer Lists

The peer list contains peer(s) that the local PCEP entity knows about. A PCEP speaker is identified by its IP address. If there is a PCEP speaker in the network that uses multiple IP addresses then it looks like multiple distinct peers to the other PCEP speakers in the network.

Since PCEP sessions can be ephemeral, the peer list tracks a peer even when no PCEP session currently exists to that peer. The statistics contained are an aggregate of the statistics for all successive sessions to that peer.

To limit the quantity of information that is stored, an implementation MAY choose to discard this information if and only if no PCEP session exists to the corresponding peer.

The data model for PCEP peer presented in this document uses a flat list of peers. Each peer in the list is identified by its IP address (addr-type, addr).

There is one list for static peer configuration ("/pcep/entity/peers"), and a separate list for the operational state of all peers (i.e. static as well as discovered)("/pcep-state/entity/peers"). The former is used to enable remote PCE configuration at PCC (or PCE) while the latter has the operational state of these peers as well as the remote PCE peer which were discovered and PCC peers that have initiated session.

5.3. The Session Lists

The session list contains PCEP session that the PCEP entity (PCE or PCC) is currently participating in. The statistics in session are semantically different from those in peer since the former applies to the current session only, whereas the latter is the aggregate for all sessions that have existed to that peer.

Although [RFC5440] forbids more than one active PCEP session between a given pair of PCEP entities at any given time, there is a window during session establishment where two sessions may exist for a given pair, one representing a session initiated by the local PCEP entity and the other representing a session initiated by the peer. If either of these sessions reaches active state first, then the other is discarded.

The data model for PCEP session presented in this document uses a flat list of sessions. Each session in the list is identified by its

initiator. This index allows two sessions to exist transiently for a given peer, as discussed above.

There is only one list for the operational state of all sessions ("/pcep-state/entity/peers/peer/sessions/session").

5.4. Notifications

This YANG model defines a list of notifications to inform client of important events detected during the protocol operation. The notifications defined cover the PCEP MIB notifications.

6. Advanced PCE Features

This document contains a specification of the base PCEP YANG module, "ietf-pcep" which provides the basic PCEP [RFC5440] data model.

This document further handles advanced PCE features like -

- Capability and Scope
- Domain information (local/neighbour)
- Path-Key
- OF
- GCO
- P2MP
- GMPLS
- Inter-Layer
- Stateful PCE
- Segement Routing
- Authentication including PCEPS (TLS)

[Editor’s Note - Some of them would be added in a future revision.]

6.1. Stateful PCE’s LSP-DB

In the operational state of PCEP which supports stateful PCE mode, the list of LSP state are maintained in LSP-DB. The key is the PLSP-ID and the PCC IP address.

The PCEP data model contains the operational state of LSPs (/pcep-state/entity/lsp-db/lsp/) with PCEP specific attributes. The generic TE attributes of the LSP are defined in [I-D.ietf-teas-yang-te]. A reference to LSP state in TE model is maintained.

7. Open Issues and Next Step

This section is added so that open issues can be tracked. This section would be removed when the document is ready for publication.

7.1. The PCE-Initiated LSP

The TE Model at [I-D.ietf-teas-yang-te] should support creationg of tunnels at the controller (PCE) and marking them as PCE-Initiated. The LSP-DB in the PCEP Yang (/pcep-state/entity/lsp-db/lsp/initiation) also marks the LSPs which are PCE-initiated.

7.2. PCEP over TLS (PCEPS)

A future version of this document would add TLS related configurations.

8. PCEP YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

```
<CODE BEGINS> file "ietf-pcep@2016-07-07.yang"
module ietf-pcep {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pcep";
    prefix pcep;

    import ietf-inet-types {
        prefix "inet";
    }

    import ietf-yang-types {
        prefix "yang";
    }

    import ietf-te {
        prefix "te";
    }

    import ietf-key-chain {
       prefix "key-chain";
```

```
    }


    organization
        "IETF PCE (Path Computation Element) Working Group";

    contact
        "WG Web:   <http://tools.ietf.org/wg/pce/>
         WG List:  <mailto:pce@ietf.org>
         WG Chair: JP Vasseur
                   <mailto:jpv@cisco.com>
         WG Chair: Julien Meuric
                   <mailto:julien.meuric@orange.com>
         WG Chair: Jonathan Hardwick
                   <mailto:Jonathan.Hardwick@metaswitch.com>
         Editor:   Dhruv Dhody
                   <mailto:dhruv.ietf@gmail.com>";

    description
        "The YANG module defines a generic configuration and
         operational model for PCEP common across all of the
         vendor implementations.";


    revision 2016-07-07 {
        description "Initial revision.";
        reference
            "RFC XXXX:  A YANG Data Model for Path Computation
                        Element Communications Protocol
                        (PCEP)";
    }

    /*
     * Identities
     */

    identity pcep {
        description "Identity for the PCEP protocol.";
    }

    /*
     * Typedefs
     */
    typedef pcep-role {
        type enumeration {
            enum unknown {
                value "0";
                description
```

```
                "An unknown role";
            }
            enum pcc {
                value "1";
                description
                "The role of a Path Computation Client";
            }
            enum pce {
                value "2";
                description
                "The role of Path Computation Element";
            }
            enum pcc-and-pce {
                value "3";
                description
                "The role of both Path Computation Client and
                 Path Computation Element";
            }
        }

        description
            "The role of a PCEP speaker.
             Takes one of the following values
             - unknown(0): the role is not known.
             - pcc(1): the role is of a Path Computation
               Client (PCC).
             - pce(2): the role is of a Path Computation
               Server (PCE).
             - pccAndPce(3): the role is of both a PCC and
               a PCE.";

    }

    typedef pcep-admin-status {
        type enumeration {
                enum admin-status-up {
                value "1";
                description
                "Admin Status is Up";
            }
            enum admin-status-down {
                value "2";
                description
                "Admin Status is Down";
            }
        }

        description
```

```
        "The Admin Status of the PCEP entity.
         Takes one of the following values
             - admin-status-up(1): Admin Status is Up.
             - admin-status-down(2): Admin Status is Down";
    }

    typedef pcep-oper-status {
        type enumeration {
            enum oper-status-up {
                value "1";
                description
                "The PCEP entity is active";
            }
            enum oper-status-down {
                value "2";
                description
                "The PCEP entity is inactive";
            }
            enum oper-status-going-up {
                value "3";
                description
                "The PCEP entity is activating";
            }
            enum oper-status-going-down {
                value "4";
                description
                "The PCEP entity is deactivating";
            }
            enum oper-status-failed {
                value "5";
                description
                "The PCEP entity has failed and will recover
                 when possible.";
            }
            enum oper-status-failed-perm {
                value "6";
                description
                "The PCEP entity has failed and will not recover
                 without operator intervention";
            }
        }
        description
        "The operational status of the PCEP entity.
         Takes one of the following values
             - oper-status-up(1): Active
             - oper-status-down(2): Inactive
             - oper-status-going-up(3): Activating
             - oper-status-going-down(4): Deactivating
```

```
             - oper-status-failed(5): Failed
             - oper-status-failed-perm(6): Failed Permanantly";
    }

    typedef pcep-initiator {
        type enumeration {
            enum local {
                value "1";
                description
                "The local PCEP entity initiated the session";
            }

            enum remote {
                value "2";
                description
                "The remote PCEP peer initiated the session";
            }
        }
        description
        "The initiator of the session, that is, whether the TCP
         connection was initiated by the local PCEP entity or
         the remote peer.
         Takes one of the following values
             - local(1): Initiated locally
             - remote(2): Initiated remotely";
    }

    typedef pcep-sess-state {
        type enumeration {
            enum tcp-pending {
                value "1";
                description
                    "The tcp-pending state of PCEP session.";
            }

            enum open-wait {
                value "2";
                description
                    "The open-wait state of PCEP session.";
            }

            enum keep-wait {
                value "3";
                description
                    "The keep-wait state of PCEP session.";
            }

            enum session-up {
```

```
                value "4";
                description
                    "The session-up state of PCEP session.";
            }
        }
        description
            "The current state of the session.
             The set of possible states excludes the idle state
             since entries do not exist in the idle state.
             Takes one of the following values
                - tcp-pending(1): PCEP TCP Pending state
                - open-wait(2): PCEP Open Wait state
                - keep-wait(3): PCEP Keep Wait state
                - session-up(4): PCEP Session Up state";
    }

    typedef domain-type {
        type enumeration {
            enum ospf-area {
                value "1";
                description
                    "The OSPF area.";
            }
            enum isis-area {
                value "2";
                description
                    "The IS-IS area.";
            }
            enum as {
                value "3";
                description
                    "The Autonomous System (AS).";
            }
        }
        description
            "The PCE Domain Type";
    }

    typedef domain-ospf-area {
        type union {
            type uint32;
            type yang:dotted-quad;
       }
       description
            "OSPF Area ID.";
    }

    typedef domain-isis-area {
```

```
        type string {
            pattern '[0-9A-Fa-f]{2}\.([0-9A-Fa-f]{4}\.){0,3}';
        }
        description
            "IS-IS Area ID.";
    }

    typedef domain-as {
        type uint32;
        description
            "Autonomous System number.";

    }

    typedef domain {
        type union {
            type domain-ospf-area;
            type domain-isis-area;
            type domain-as;
        }
        description
            "The Domain Information";
    }

    typedef operational-state {
        type enumeration {
            enum down {
                value "0";
                description
                    "not active.";
            }
            enum up {
                value "1";
                description
                    "signalled.";
            }
            enum active {
                value "2";
                description
                    "up and carrying traffic.";
            }
            enum going-down {
                value "3";
                description
                    "LSP is being torn down, resources are
                     being released.";
            }
            enum going-up {
```

```
                value "4";
                description
                    "LSP is being signalled.";
            }
        }
        description
            "The operational status of the LSP";
    }

    typedef lsp-error {
        type enumeration {
            enum no-error {
                value "0";
                description
                    "No error, LSP is fine.";
            }
            enum unknown {
                value "1";
                description
                    "Unknown reason.";
            }
            enum limit {
                value "2";
                description
                    "Limit reached for PCE-controlled LSPs.";
            }
            enum pending {
                value "3";
                description
                    "Too many pending LSP update requests.";
            }
            enum unacceptable {
                value "4";
                description
                    "Unacceptable parameters.";
            }
            enum internal {
                value "5";
                description
                    "Internal error.";
            }
            enum admin {
                value "6";
                description
                    "LSP administratively brought down.";
            }
            enum preempted {
                value "7";
```

```
                description
                    "LSP preempted.";
            }
            enum rsvp {
                value "8";
                description
                    "RSVP signaling error.";
            }
        }
        description
            "The LSP Error Codes.";
    }

    typedef sync-state {
        type enumeration {
            enum pending {
                value "0";
                description
                    "The state synchronization
                     has not started.";
            }
            enum ongoing {
                value "1";
                description
                    "The state synchronization
                     is ongoing.";
            }
            enum finished {
                value "2";
                description
                    "The state synchronization
                     is finished.";
            }
        }
        description
            "The LSP-DB state synchronization operational status.";
    }

        typedef pst{
                type enumeration{
                        enum rsvp-te{
                                value "0";
                                description
                                        "RSVP-TE signaling protocol";
                        }
                        enum sr{
                                value "1";
                                description
```

```
                                        "Segment Routing Traffic Engineering";
                        }
                }
                description
                        "The Path Setup Type";
        }

    typedef assoc-type{
        type enumeration{
            enum protection{
                value "1";
                description
                    "Path Protection Association Type";
            }
        }
        description
            "The PCEP Association Type";
    }

    /*
     * Features
     */

    feature svec {
        description
            "Support synchronized path computation.";
    }

    feature gmpls {
        description
            "Support GMPLS.";
    }

    feature obj-fn {
        description
            "Support OF as per RFC 5541.";
    }

    feature gco {
        description
            "Support GCO as per RFC 5557.";
    }

    feature pathkey {
        description
            "Support pathkey as per RFC 5520.";
    }
```

```
    feature p2mp {
        description
            "Support P2MP as per RFC 6006.";
    }

    feature stateful {
        description
            "Support stateful PCE.";
    }

    feature pce-initiated {
        description
            "Support PCE-Initiated LSP.";
    }

    feature tls {
        description
            "Support PCEP over TLS.";
    }

    feature sr {
        description
            "Support Segement Routing for PCE.";
    }

    /*
     * Groupings
     */



    grouping pcep-entity-info{
        description
            "This grouping defines the attributes for PCEP entity.";
        leaf connect-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
                "The time in seconds that the PCEP entity will wait
                 to establish a TCP connection with a peer.  If a
                 TCP connection is not established within this time
                 then PCEP aborts the session setup attempt.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
```

```
        }

        leaf connect-max-retry {
            type uint32;
            default 5;
            description
                "The maximum number of times the system tries to
                 establish a TCP connection to a peer before the
                 session with the peer transitions to the idle
                 state.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf init-backoff-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
               "The initial back-off time in seconds for retrying
                a failed session setup attempt to a peer.
                The back-off time increases for each failed
                session setup attempt, until a maximum back-off
                time is reached.  The maximum back-off time is
                max-backoff-timer.";
        }

        leaf max-backoff-timer {
            type uint32;
            units "seconds";
            description
               "The maximum back-off time in seconds for retrying
                a failed session setup attempt to a peer.
                The back-off time increases for each failed session
                setup attempt, until this maximum value is reached.
                Session setup attempts then repeat periodically
                without any further increase in back-off time.";
        }

        leaf open-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
```

```
               "The time in seconds that the PCEP entity will wait
                to receive an Open message from a peer after the
                TCP connection has come up.
                If no Open message is received within this time then
                PCEP terminates the TCP connection and deletes the
                associated sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf keep-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
                "The time in seconds that the PCEP entity will wait
                 to receive a Keepalive or PCErr message from a peer
                 during session initialization after receiving an
                 Open message.  If no Keepalive or PCErr message is
                 received within this time then PCEP terminates the
                 TCP connection and deletes the associated
                 sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf keep-alive-timer {
            type uint32 {
                range "0..255";
            }
            units "seconds";
            default 30;
            description
                "The keep alive transmission timer that this PCEP
                 entity will propose in the initial OPEN message of
                 each session it is involved in.  This is the
                 maximum time between two consecutive messages sent
                 to a peer. Zero means that the PCEP entity prefers
                 not to send Keepalives at all.
                 Note that the actual Keepalive transmission
                 intervals, in either direction of an active PCEP
                 session, are determined by negotiation between the
                 peers as specified by RFC 5440, and so may differ
                 from this configured value.";
```

```
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf dead-timer {
            type uint32 {
                range "0..255";
            }
            units "seconds";
            must ". >= ../keep-alive-timer" {
                error-message "The dead timer must be "
                       + "larger than the keep alive timer";
                description
                    "This value MUST be greater than
                     keep-alive-timer.";

            }
            default 120;
            description
                "The dead timer that this PCEP entity will propose
                 in the initial OPEN message of each session it is
                 involved in. This is the time after which a peer
                 should declare a session down if it does not
                 receive any PCEP messages. Zero suggests that the
                 peer does not run a dead timer at all." ;
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf allow-negotiation{
            type boolean;
            description
                "Whether the PCEP entity will permit negotiation of
                 session parameters.";
        }

        leaf max-keep-alive-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the maximum value that this PCEP entity will
                 accept from a peer for the interval between
                 Keepalive transmissions. Zero means that the PCEP
```

```
                entity will allow no Keepalive transmission at
                all." ;
        }

        leaf max-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the maximum value that this PCEP entity will accept
                 from a peer for the Dead timer.  Zero means that
                 the PCEP entity will allow not running a Dead
                 timer.";
        }

        leaf min-keep-alive-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the minimum value that this PCEP entity will
                 accept for the interval between Keepalive
                 transmissions. Zero means that the PCEP entity
                 insists on no Keepalive transmission at all.";
        }

        leaf min-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                 "In PCEP session parameter negotiation in
                  seconds, the minimum value that this PCEP entity
                  will accept for the Dead timer.  Zero means that
                  the PCEP entity insists on not running a Dead
                  timer.";
        }

        leaf sync-timer{
            if-feature svec;
            type uint32 {
                range "0..65535";
            }
```

```
            units "seconds";
            default 60;
            description
                "The value of SyncTimer in seconds is used in the
                 case of synchronized path computation request
                 using the SVEC object. Consider the case where a
                 PCReq message is received by a PCE that contains
                 the SVEC object referring to M synchronized path
                 computation requests.  If after the expiration of
                 the SyncTimer all the M path computation requests
                 have not been, received a protocol error is
                 triggered and the PCE MUST cancel the whole set
                 of path computation  requests.
                 The aim of the SyncTimer is to avoid the storage
                 of unused synchronized requests should one of
                 them get lost for some reasons (for example, a
                 misbehaving PCC).
                 Zero means that the PCEP entity does not use the
                 SyncTimer.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }


        leaf request-timer{
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
                "The maximum time that the PCEP entity will wait
                 for a response to a PCReq message.";
        }

        leaf max-sessions{
            type uint32;
            description
               "Maximum number of sessions involving this PCEP
                entity that can exist at any time.";
        }

        leaf max-unknown-reqs{
            type uint32;
            default 5;
            description
              "The maximum number of unrecognized requests and
               replies that any session on this PCEP entity is
```

```
                willing to accept per minute before terminating
                the session.
                A PCRep message contains an unrecognized reply
                if it contains an RP object whose request ID
                does not correspond to any in-progress request
                sent by this PCEP entity.
                A PCReq message contains an unrecognized request
                if it contains an RP object whose request ID is
                zero.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf max-unknown-msgs{
            type uint32;
            default 5;
            description
             "The maximum number of unknown messages that any
              session on this PCEP entity is willing to accept
              per minute before terminating the session.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

    }//pcep-entity-info

    grouping  pce-scope{
        description
            "This grouping defines PCE path computation scope
             information which maybe relevant to PCE selection.
             This information corresponds to PCE auto-discovery
             information.";
        reference
            "RFC 5088: OSPF Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery
             RFC 5089: IS-IS Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery";
        leaf intra-area-scope{
            type boolean;
            default true;
            description
                "PCE can compute intra-area paths.";
        }
        leaf intra-area-pref{
```

```
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for intra-area TE LSP
              computation.";
        }
        leaf inter-area-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-area paths.";
        }
        leaf inter-area-scope-default{
            type boolean;
            default false;
            description
                "PCE can act as a default PCE for inter-area
                 path computation.";
        }
        leaf inter-area-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-area TE LSP
              computation.";
        }
        leaf inter-as-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-AS paths.";
        }
        leaf inter-as-scope-default{
            type boolean;
            default false;
            description
                "PCE can act as a default PCE for inter-AS
                 path computation.";
        }
        leaf inter-as-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-AS TE LSP
              computation.";
```

```
        }
        leaf inter-layer-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-layer paths.";
        }
        leaf inter-layer-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-layer TE LSP
              computation.";
        }
    }//pce-scope

    grouping  domain{
        description
            "This grouping specifies a Domain where the
             PCEP speaker has topology visibility.";
        leaf domain-type{
            type domain-type;
            description
              "The domain type.";
        }
        leaf domain{
            type domain;
            description
              "The domain Information.";
        }
    }//domain

    grouping capability{
        description
            "This grouping specifies a capability
             information of local PCEP entity. This maybe
             relevant to PCE selection as well. This
             information corresponds to PCE auto-discovery
             information.";
        reference
            "RFC 5088: OSPF Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery
             RFC 5089: IS-IS Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery";
        leaf gmpls{
```

```
            if-feature gmpls;
            type boolean;
            description
              "Path computation with GMPLS link
               constraints.";
          }
          leaf bi-dir{
            type boolean;
            description
              "Bidirectional path computation.";
          }
          leaf diverse{
            type boolean;
            description
              "Diverse path computation.";
          }
          leaf load-balance{
            type boolean;
            description
              "Load-balanced path computation.";
          }
          leaf synchronize{
            if-feature svec;
            type boolean;
            description
              "Synchronized paths computation.";
          }
          leaf objective-function{
            if-feature obj-fn;
            type boolean;
            description
              "Support for multiple objective functions.";
          }
          leaf add-path-constraint{
            type boolean;
            description
              "Support for additive path constraints (max
               hop count, etc.).";
          }
          leaf prioritization{
            type boolean;
            description
              "Support for request prioritization.";
          }
          leaf multi-request{
            type boolean;
            description
              "Support for multiple requests per message.";
```

```
        }
        leaf gco{
            if-feature gco;
            type boolean;
            description
              "Support for Global Concurrent Optimization
               (GCO).";
        }
        leaf p2mp{
            if-feature p2mp;
            type boolean;
            description
              "Support for P2MP path computation.";
        }

        container stateful{
            if-feature stateful;
            description
                "If stateful PCE feature is present";
            leaf enabled{
                type boolean;
                description
                    "Enabled or Disabled";
            }
            leaf active{
                type boolean;
                description
                  "Support for active stateful PCE.";
            }
            leaf pce-initiated{
                if-feature pce-initiated;
                type boolean;
                description
                  "Support for PCE-initiated LSP.";
            }
        }
                container sr{
                        if-feature sr;
                        description
                                "If segment routing is supported";
                        leaf enabled{
                type boolean;
                description
                    "Enabled or Disabled";
                        }
                        leaf msd{ /*should be in MPLS yang model (?)*/
                                type uint8;
                must "((../../role == 'pcc')" +
```

```
                     " or " +
                    "(../../role == 'pcc-and-pce')))"
                {
                    error-message
                        "The PCEP entity must be PCC";
                    description
                        "When PCEP entity is PCC for
                        MSD to be applicable";
                }
                                description
                                        "Maximum SID Depth";
                        }
                }
    }//capability

    grouping  info{
        description
            "This grouping specifies all information which
             maybe relevant to both PCC and PCE.
             This information corresponds to PCE auto-discovery
             information.";
        container domain{
            description
                "The local domain for the PCEP entity";
            list domain{
                key "domain-type domain";
                description
                    "The local domain.";
                uses domain{
                    description
                        "The local domain for the PCEP entity.";
                }
            }
        }
        container capability{
            description
                "The PCEP entity capability";
            uses capability{
                description
                    "The PCEP entity supported
                    capabilities.";
            }
        }
    }//info

    grouping  pce-info{
        description
            "This grouping specifies all PCE information
```

```
             which maybe relevant to the PCE selection.
             This information corresponds to PCE auto-discovery
             information.";
        container scope{
            description
                "The path computation scope";
            uses pce-scope;
        }

        container neigh-domains{
            description
                "The list of neighbour PCE-Domain
                 toward which a PCE can compute
                 paths";
            list domain{
                key "domain-type domain";

                description
                    "The neighbour domain.";
                uses domain{
                    description
                        "The PCE neighbour domain.";
                }
            }
        }
    }//pce-info

    grouping  pcep-stats{
        description
            "This grouping defines statistics for PCEP. It is used
             for both peer and current session.";
        leaf avg-rsp-time{
            type uint32;
            units "milliseconds";
            must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
                 " or " +
                 "(/pcep-state/entity/peers/peer/role = 'pcc'" +
                 " and avg-rsp-time = 0))" {
                error-message
                    "Invalid average response time";
                description
                    "If role is pcc then this leaf is meaningless
                     and is set to zero.";
            }
            description
              "The average response time.
               If an average response time has not been
               calculated then this leaf has the value zero.";
```

```
        }

        leaf lwm-rsp-time{
            type uint32;
            units "milliseconds";
            must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
                 " or " +
                 "(/pcep-state/entity/peers/peer/role = 'pcc'" +
                 " and lwm-rsp-time = 0))" {
                error-message
                    "Invalid smallest (low-water mark)
                     response time";
                description
                    "If role is pcc then this leaf is meaningless
                     and is set to zero.";
            }
            description
             "The smallest (low-water mark) response time seen.
              If no responses have been received then this
              leaf has the value zero.";
        }

        leaf hwm-rsp-time{
            type uint32;
            units "milliseconds";
            must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
                 " or " +
                 "(/pcep-state/entity/peers/peer/role = 'pcc'" +
                 " and hwm-rsp-time = 0))" {
                error-message
                    "Invalid greatest (high-water mark)
                     response time seen";
                description
                    "If role is pcc then this field is
                     meaningless and is set to zero.";
            }
            description
             "The greatest (high-water mark) response time seen.
              If no responses have been received then this object
              has the value zero.";
        }

        leaf num-pcreq-sent{
            type yang:counter32;
            description
             "The number of PCReq messages sent.";
        }
```

```
        leaf num-pcreq-rcvd{
            type yang:counter32;
            description
              "The number of PCReq messages received.";
        }

        leaf num-pcrep-sent{
            type yang:counter32;
            description
              "The number of PCRep messages sent.";
        }

        leaf num-pcrep-rcvd{
            type yang:counter32;
            description
              "The number of PCRep messages received.";
        }

        leaf num-pcerr-sent{
            type yang:counter32;
            description
              "The number of PCErr messages sent.";
        }

        leaf num-pcerr-rcvd{
            type yang:counter32;
            description
              "The number of PCErr messages received.";
        }

        leaf num-pcntf-sent{
            type yang:counter32;
            description
              "The number of PCNtf messages sent.";
        }

        leaf num-pcntf-rcvd{
            type yang:counter32;
            description
              "The number of PCNtf messages received.";
        }

        leaf num-keepalive-sent{
            type yang:counter32;
            description
              "The number of Keepalive messages sent.";
        }
```

```
        leaf num-keepalive-rcvd{
            type yang:counter32;
            description
              "The number of Keepalive messages received.";
        }

        leaf num-unknown-rcvd{
            type yang:counter32;
            description
              "The number of unknown messages received.";
        }

        leaf num-corrupt-rcvd{
            type yang:counter32;
            description
              "The number of corrupted PCEP message received.";
        }

        leaf num-req-sent{
            type yang:counter32;
            description
              "The number of requests sent.  A request corresponds
               1:1 with an RP object in a PCReq message. This might
               be greater than num-pcreq-sent because multiple
               requests can be batched into a single PCReq
               message.";
        }

        leaf num-req-sent-pend-rep{
            type yang:counter32;
            description
              "The number of requests that have been sent for
               which a response is still pending.";
        }

        leaf num-req-sent-ero-rcvd{
            type yang:counter32;
            description
               "The number of requests that have been sent for
                which a response with an ERO object was received.
                Such responses indicate that a path was
                successfully computed by the peer.";
        }

        leaf num-req-sent-nopath-rcvd{
            type yang:counter32;
            description
              "The number of requests that have been sent for
```

```
            which a response with a NO-PATH object was
            received. Such responses indicate that the peer
            could not  find a path to satisfy the
            request.";
        }

        leaf num-req-sent-cancel-rcvd{
            type yang:counter32;
            description
              "The number of requests that were cancelled with
               a PCNtf message.
               This might be different than num-pcntf-rcvd because
               not all PCNtf messages are used to cancel requests,
               and a single PCNtf message can cancel multiple
               requests.";
        }

        leaf num-req-sent-error-rcvd{
            type yang:counter32;
            description
             "The number of requests that were rejected with a
              PCErr message.
              This might be different than num-pcerr-rcvd because
              not all PCErr messages are used to reject requests,
              and a single PCErr message can reject multiple
              requests.";
        }

        leaf num-req-sent-timeout{
            type yang:counter32;
            description
             "The number of requests that have been sent to a peer
             and have been abandoned because the peer has taken too
             long to respond to them.";
        }

        leaf num-req-sent-cancel-sent{
            type yang:counter32;
            description
             "The number of requests that were sent to the peer and
              explicitly cancelled by the local PCEP entity sending
              a PCNtf.";
        }

        leaf num-req-rcvd{
            type yang:counter32;
            description
             "The number of requests received.  A request
```

```
              corresponds 1:1 with an RP object in a PCReq
              message.
              This might be greater than num-pcreq-rcvd because
              multiple requests can be batched into a single
              PCReq message.";
        }

        leaf num-req-rcvd-pend-rep{
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response is still pending.";
        }

        leaf num-req-rcvd-ero-sent{
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response with an ERO object was sent.  Such
              responses indicate that a path was successfully
              computed by the local PCEP entity.";
        }

        leaf num-req-rcvd-nopath-sent{
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response with a NO-PATH object was sent. Such
              responses indicate that the local PCEP entity could
              not find a path to satisfy the request.";
        }

        leaf num-req-rcvd-cancel-sent{
            type yang:counter32;
            description
             "The number of requests received that were cancelled
              by the local PCEP entity sending a PCNtf message.
              This might be different than num-pcntf-sent because
              not all PCNtf messages are used to cancel requests,
              and a single PCNtf message can cancel multiple
              requests.";
        }

        leaf num-req-rcvd-error-sent{
            type yang:counter32;
            description
             "The number of requests received that were cancelled
              by the local PCEP entity sending a PCErr message.
```

```
              This might be different than num-pcerr-sent because
              not all PCErr messages are used to cancel requests,
              and a single PCErr message can cancel multiple
              requests.";
        }

        leaf  num-req-rcvd-cancel-rcvd{
            type yang:counter32;
            description
             "The number of requests that were received from the
              peer and explicitly cancelled by the peer sending
              a PCNtf.";
        }

        leaf  num-rep-rcvd-unknown{
            type yang:counter32;
            description
              "The number of responses to unknown requests
               received. A response to an unknown request is a
               response whose RP object does not contain the
               request ID of any request that is currently
               outstanding on the session.";
        }

        leaf  num-req-rcvd-unknown{
            type yang:counter32;
            description
              "The number of unknown requests that have been
               received. An unknown request is a request
               whose RP object contains a request ID of
               zero.";
        }

        container svec{
            if-feature svec;
            description
                "If synchronized path computation is supported";
            leaf num-svec-sent{
                type yang:counter32;
                description
                  "The number of SVEC objects sent in PCReq messages.
                   An SVEC object represents a set of synchronized
                   requests.";
            }

            leaf num-svec-req-sent{
                type yang:counter32;
                description
```

```
                  "The number of requests sent that appeared in one
                   or more SVEC objects.";
            }

            leaf num-svec-rcvd{
                type yang:counter32;
                description
                 "The number of SVEC objects received in PCReq
                  messages. An SVEC object represents a set of
                  synchronized requests.";
            }

            leaf num-svec-req-rcvd{
                type yang:counter32;
                description
                  "The number of requests received that appeared
                   in one or more SVEC objects.";
            }
        }
        container stateful{
            if-feature stateful;
            description
                "Stateful PCE related statistics";
            leaf num-pcrpt-sent{
                type yang:counter32;
                description
                 "The number of PCRpt messages sent.";
            }

            leaf num-pcrpt-rcvd{
                type yang:counter32;
                description
                  "The number of PCRpt messages received.";
            }

            leaf num-pcupd-sent{
                type yang:counter32;
                description
                 "The number of PCUpd messages sent.";
            }

            leaf num-pcupd-rcvd{
                type yang:counter32;
                description
                  "The number of PCUpd messages received.";
            }

            leaf num-rpt-sent{
```

```
                type yang:counter32;
                description
                  "The number of LSP Reports sent.  A LSP report
                   corresponds 1:1 with an LSP object in a PCRpt
                   message. This might be greater than
                   num-pcrpt-sent because multiple reports can
                   be batched into a single PCRpt message.";
            }

            leaf num-rpt-rcvd{
                type yang:counter32;
                description
                 "The number of LSP Reports received.  A LSP report
                  corresponds 1:1 with an LSP object in a PCRpt
                  message.
                  This might be greater than num-pcrpt-rcvd because
                  multiple reports can be batched into a single
                  PCRpt message.";
            }

            leaf  num-rpt-rcvd-error-sent{
                type yang:counter32;
                description
                  "The number of reports of LSPs received that were
                  responded by the local PCEP entity by sending a
                  PCErr message.";
            }

            leaf num-upd-sent{
                type yang:counter32;
                description
                  "The number of LSP updates sent.  A LSP update
                   corresponds 1:1 with an LSP object in a PCUpd
                   message. This might be greater than
                   num-pcupd-sent because multiple updates can
                   be batched into a single PCUpd message.";
            }

            leaf num-upd-rcvd{
                type yang:counter32;
                description
                 "The number of LSP Updates received.  A LSP update
                  corresponds 1:1 with an LSP object in a PCUpd
                  message.
                  This might be greater than num-pcupd-rcvd because
                  multiple updates can be batched into a single
                  PCUpd message.";
            }
```

```
            leaf  num-upd-rcvd-unknown{
                type yang:counter32;
                description
                  "The number of updates to unknown LSPs
                   received. An update to an unknown LSP is a
                   update whose LSP object does not contain the
                   PLSP-ID of any LSP that is currently
                   present.";
            }

            leaf  num-upd-rcvd-undelegated{
                type yang:counter32;
                description
                  "The number of updates to not delegated LSPs
                   received. An update to an undelegated LSP is a
                   update whose LSP object does not contain the
                   PLSP-ID of any LSP that is currently
                   delegated to current PCEP session.";
            }

            leaf  num-upd-rcvd-error-sent{
                type yang:counter32;
                description
                  "The number of updates to LSPs received that were
                  responded by the local PCEP entity by sending a
                  PCErr message.";
            }
            container initiation {
                if-feature pce-initiated;
                description
                    "PCE-Initiated related statistics";
                leaf num-pcinitiate-sent{
                    type yang:counter32;
                    description
                     "The number of PCInitiate messages sent.";
                }

                leaf num-pcinitiate-rcvd{
                    type yang:counter32;
                    description
                      "The number of PCInitiate messages received.";
                }

                leaf num-initiate-sent{
                    type yang:counter32;
                    description
                      "The number of LSP Initiation sent via PCE.
                       A LSP initiation corresponds 1:1 with an LSP
```

```
                        object in a PCInitiate message. This might be
                        greater than num-pcinitiate-sent because
                        multiple initiations can be batched into a
                        single PCInitiate message.";
                }

                leaf num-initiate-rcvd{
                    type yang:counter32;
                    description
                      "The number of LSP Initiation received from
                       PCE.  A LSP initiation corresponds 1:1 with
                       an LSP object in a PCInitiate message. This
                       might be greater than num-pcinitiate-rcvd
                       because multiple initiations can be batched
                       into a single PCInitiate message.";
                }

                leaf  num-initiate-rcvd-error-sent{
                    type yang:counter32;
                    description
                      "The number of initiations of LSPs received
                       that were responded by the local PCEP entity
                       by sending a PCErr message.";
                }
            }
        }
    }//pcep-stats

    grouping lsp-state{
        description
            "This grouping defines the attributes for LSP in LSP-DB.
             These are the attributes specifically from the PCEP
             perspective";
        leaf plsp-id{
            type uint32{
                range "1..1048575";
            }
            description
                "A PCEP-specific identifier for the LSP.  A PCC
                 creates a unique PLSP-ID for each LSP that is
                 constant for the lifetime of a PCEP session.
                 PLSP-ID is 20 bits with 0 and 0xFFFFF are
                 reserved";
        }
        leaf pcc-id{
            type inet:ip-address;
            description
                "The local internet address of the PCC, that
```

```
                generated the PLSP-ID.";
        }

        container lsp-ref{
            description
                "reference to ietf-te lsp state";

            leaf source {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:source";
                }
                description
                  "Tunnel sender address extracted from
                  SENDER_TEMPLATE  object";
                reference "RFC3209";
            }
            leaf destination {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:"
                         + "destination";
                }
                description
                    "Tunnel endpoint address extracted from
                    SESSION object";
                reference "RFC3209";
            }
            leaf tunnel-id {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:tunnel-id";
                }
                description
                    "Tunnel identifier used in the SESSION
                    that remains constant over the life
                    of the tunnel.";
                reference "RFC3209";
            }
            leaf lsp-id {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:lsp-id";
                }
                description
                    "Identifier used in the SENDER_TEMPLATE
                    and the FILTER_SPEC that can be changed
                    to allow a sender to share resources with
                    itself.";
                reference "RFC3209";
            }
            leaf extended-tunnel-id {
```

```
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:"
                    + "extended-tunnel-id";
                }
                description
                    "Extended Tunnel ID of the LSP.";
                reference "RFC3209";
            }
            leaf type {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:type";
                }
                description "LSP type P2P or P2MP";
            }
        }

        leaf admin-state{
            type boolean;
            description
                "The desired operational state";
        }
        leaf operational-state{
            type operational-state;
            description
                "The operational status of the LSP";
        }
        container delegated{
            description
                "The delegation related parameters";
            leaf enabled{
                type boolean;
                description
                    "LSP is delegated or not";
            }
            leaf pce{
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                must "((../enabled == true)" +
                     " and " +
                     "((../../role == 'pcc')" +
                     " or " +
                    "(../../role == 'pcc-and-pce')))"
                {
                    error-message
                        "The PCEP entity must be PCC
                         and the LSP be delegated";
                    description
```

```
                    "When PCEP entity is PCC for
                    delegated LSP";
                }
                description
                    "The reference to the PCE peer to
                    which LSP is delegated";
            }
            leaf srp-id{
                type uint32;
                description
                    "The last SRP-ID-number associated with this
                    LSP.";
            }
        }
        container initiation {
            if-feature pce-initiated;
            description
                "The PCE initiation related parameters";
            leaf enabled{
                type boolean;
                description
                    "LSP is PCE-initiated or not";
            }
            leaf pce{
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                must "(../enabled == true)"
                {
                    error-message
                        "The LSP must be PCE-Initiated";
                    description
                        "When the LSP must be PCE-Initiated";
                }
                description
                    "The reference to the PCE
                    that initiated this LSP";
            }
        }
        leaf symbolic-path-name{
            type string;
            description
                "The symbolic path name associated with the LSP.";
        }
        leaf last-error{
            type lsp-error;
            description
                "The last error for the LSP.";
```

```
        }
                leaf pst{
                        type pst;
                        default "rsvp-te";
                        description
                                "The Path Setup Type";

                }

    }//lsp-state

    grouping notification-instance-hdr {
        description
            "This group describes common instance specific data
             for notifications.";

        leaf peer-addr {
            type leafref {
                path "/pcep-state/entity/peers/peer/addr";
            }
            description
                "Reference to peer address";
        }

    }// notification-instance-hdr

    grouping notification-session-hdr {
        description
        "This group describes common session instance specific
         data for notifications.";

        leaf session-initiator {
            type leafref {
            path "/pcep-state/entity/peers/peer/sessions/" +
                 "session/initiator";
            }
            description
                "Reference to pcep session initiator leaf";
        }
    }// notification-session-hdr

    grouping stateful-pce-parameter {
        description
        "This group describes stateful PCE specific
         parameters.";
        leaf state-timeout{
            type uint32;
            units "seconds";
```

```
            description
                "When a PCEP session is terminated, a PCC
                 waits for this time period before flushing
                 LSP state associated with that PCEP session
                 and reverting to operator-defined default
                 parameters or behaviours.";
        }
        leaf redelegation-timeout{
            type uint32;
            units "seconds";
            must "((../role == 'pcc')" +
                 " or " +
                 "(../role == 'pcc-and-pce'))"
            {
                error-message "The PCEP entity must be PCC";
                description
                    "When PCEP entity is PCC";
            }
            description
                "When a PCEP session is terminated, a PCC
                 waits for this time period before revoking
                 LSP delegation to a PCE and attempting to
                 redelegate LSPs associated with the
                 terminated PCEP session to an alternate
                 PCE.";
        }
        leaf rpt-non-pcep-lsp{
            type boolean;
            must "((../role == 'pcc')" +
                 " or " +
                 "(../role == 'pcc-and-pce'))"
            {
                error-message "The PCEP entity must be PCC";
                description
                    "When PCEP entity is PCC";
            }
            description
                "If set, a PCC reports LSPs that are not
                controlled by any PCE (for example, LSPs
                that are statically configured at the
                PCC). ";
        }

    }

    grouping authentication {
        description "Authentication Information";
        choice auth-type-selection {
```

```
            description
                "Options for expressing authentication setting.";
            case auth-key-chain {
                leaf key-chain {
                    type key-chain:key-chain-ref;
                    description
                        "key-chain name.";
                }
            }
            case auth-key {
                leaf key {
                    type string;
                description
                    "Key string in ASCII format.";
                }
                container crypto-algorithm {
                    uses key-chain:crypto-algorithm-types;
                        description
                            "Cryptographic algorithm associated
                             with key.";
                }
            }
            case auth-tls {
                if-feature tls;
                container tls {
                    description
                        "TLS related information - TBD";
                }
            }
        }
    }

    grouping association {
        description
            "Generic Association parameters";
        leaf type {
            type "assoc-type";
            description
                "The PCEP association type";
        }
        leaf id {
            type uint16;
            description
                "PCEP Association ID";
        }
        leaf source {
          type inet:ip-address;
          description
```

```
                "PCEP Association Source.";
        }
        leaf global-source {
          type uint32;
          description
                "PCEP Association Global
                Source.";
        }
        leaf extended-id{
            type string;
            description
                "Additional information to
                support unique identification.";
        }
    }
    grouping association-ref {
        description
            "Generic Association parameters";
        leaf id {
            type leafref {
                path "/pcep-state/entity/lsp-db/"
                + "association-list/id";
            }
            description
                "PCEP Association ID";
        }
        leaf source {
            type leafref {
                path "/pcep-state/entity/lsp-db/"
                + "association-list/source";
            }
          description
                "PCEP Association Source.";
        }
        leaf global-source {
            type leafref {
                path "/pcep-state/entity/lsp-db/"
                + "association-list/global-source";
            }
          description
                "PCEP Association Global
                Source.";
        }
        leaf extended-id{
            type leafref {
                path "/pcep-state/entity/lsp-db/"
                + "association-list/extended-id";
            }
```

```
            description
                "Additional information to
                support unique identification.";
        }
    }
    /*
     * Configuration data nodes
     */
    container pcep{

        presence
            "The PCEP is enabled";

            description
            "Parameters for list of configured PCEP entities
             on the device.";

        container entity {

            description
                "The configured PCEP entity on the device.";

            leaf addr {
                type inet:ip-address;
                mandatory true;
                description
                    "The local Internet address of this PCEP
                    entity.
                    If operating as a PCE server, the PCEP
                    entity listens on this address.
                    If operating as a PCC, the PCEP entity
                    binds outgoing TCP connections to this
                    address.
                    It is possible for the PCEP entity to
                    operate both as a PCC and a PCE Server, in
                    which case it uses this address both to
                    listen for incoming TCP connections and to
                    bind outgoing TCP connections.";
            }

            leaf enabled {
                type boolean;
                default true;
                description
                    "The administrative status of this PCEP
                     Entity.";
            }
```

```
            leaf role {
                type pcep-role;
                mandatory true;
                description
                    "The role that this entity can play.
                     Takes one of the following values.
                     - unknown(0): this PCEP Entity role is not
                       known.
                     - pcc(1): this PCEP Entity is a PCC.
                     - pce(2): this PCEP Entity is a PCE.
                     - pcc-and-pce(3): this PCEP Entity is both
                       a PCC and a PCE.";
            }

            leaf description {
                type string;
                description
                    "Description of the PCEP entity configured
                     by the user";
            }

            uses info {
                description
                    "Local PCEP entity information";
            }


            container pce-info {
                must "((../role == 'pce')" +
                     " or " +
                     "(../role == 'pcc-and-pce'))"
                {
                    error-message "The PCEP entity must be PCE";
                    description
                        "When PCEP entity is PCE";
                }
                uses pce-info {
                    description
                        "Local PCE information";
                }
                                uses authentication {
                                        description
                                                "Local PCE authentication inform
ation";
                                }


                description
                    "The Local PCE Entity PCE information";
```

```
            }

            uses pcep-entity-info {
                description
                    "The configuration related to the PCEP
                     entity.";
            }



            leaf pcep-notification-max-rate {
                type uint32;
                mandatory true;
                description
                    "This variable indicates the maximum number of
                     notifications issued per second. If events occur
                     more rapidly, the implementation may simply fail
                     to emit these notifications during that period,
                     or may queue them until an appropriate time. A
                     value of 0 means no notifications are emitted
                     and all should be discarded (that is, not
                     queued).";
            }

            container stateful-parameter{
                if-feature stateful;
                must "(../info/capability/stateful/active == true)"
                {
                    error-message
                        "The Active Stateful PCE must be enabled";
                    description
                        "When PCEP entity is active stateful
                         enabled";
                }
                uses stateful-pce-parameter;

                description
                    "The configured stateful parameters";
            }


            container peers{
                must "((../role == 'pcc')" +
                     " or " +
                     "(../role == 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP entity must be PCC";
```

```
                    description
                        "When PCEP entity is PCC, as remote
                         PCE peers are configured.";
                }
                description
                    "The list of configured peers for the
                     entity (remote PCE)";
                list peer{
                    key "addr";

                    description
                        "The peer configured for the entity.
                         (remote PCE)";

                    leaf addr {
                        type inet:ip-address;
                        description
                            "The local Internet address of this
                             PCEP peer.";
                    }

                    leaf description {
                        type string;
                        description
                            "Description of the PCEP peer
                             configured by the user";
                    }
                    uses info {
                        description
                            "PCE Peer information";
                    }
                    uses pce-info {
                        description
                            "PCE Peer information";
                    }

                    leaf delegation-pref{
                        if-feature stateful;
                        type uint8{
                            range "0..7";
                        }
                        must "(../../info/capability/stateful/active"
                             + "== true)"
                        {
                            error-message
                                "The Active Stateful PCE must be
                                 enabled";
                            description
```

```
                                "When PCEP entity is active stateful
                                 enabled";
                            }
                            description
                                "The PCE peer delegation preference.";
                        }
                        uses authentication {
                            description
                                "PCE Peer authentication";
                        }
                    }//peer
                }//peers
            }//entity
        }//pcep

        /*
         * Operational data nodes
         */

        container pcep-state{
            config false;
            description
                "The list of operational PCEP entities on the
                 device.";

            container entity{
                description
                    "The operational PCEP entity on the device.";

                leaf addr {
                    type inet:ip-address;
                    description
                        "The local Internet address of this PCEP
                        entity.
                        If operating as a PCE server, the PCEP
                        entity listens on this address.
                        If operating as a PCC, the PCEP entity
                        binds outgoing TCP connections to this
                        address.
                        It is possible for the PCEP entity to
                        operate both as a PCC and a PCE Server, in
                        which case it uses this address both to
                        listen for incoming TCP connections and to
                        bind outgoing TCP connections.";
                }

                leaf index{
                    type uint32;
```

```
            description
                "The index of the operational PECP
                 entity";
        }


        leaf admin-status {
            type pcep-admin-status;
            description
                "The administrative status of this PCEP Entity.
                 This is the desired operational status as
                 currently set by an operator or by default in
                 the implementation.  The value of enabled
                 represents the current status of an attempt
                 to reach this desired status.";
        }

        leaf oper-status {
            type pcep-admin-status;
            description
               "The operational status of the PCEP entity.
                Takes one of the following values.
                - oper-status-up(1): the PCEP entity is
                  active.
                - oper-status-down(2): the PCEP entity is
                  inactive.
                - oper-status-going-up(3): the PCEP entity is
                  activating.
                - oper-status-going-down(4): the PCEP entity is
                  deactivating.
                - oper-status-failed(5): the PCEP entity has
                  failed and will recover when possible.
                - oper-status-failed-perm(6): the PCEP entity
                  has failed and will not recover without
                  operator intervention.";
        }

        leaf role {
            type pcep-role;
            description
                "The role that this entity can play.
                 Takes one of the following values.
                 - unknown(0): this PCEP entity role is
                   not known.
                 - pcc(1): this PCEP entity is a PCC.
                 - pce(2): this PCEP entity is a PCE.
                 - pcc-and-pce(3): this PCEP entity is
                   both a PCC and a PCE.";
```

```
            }

            uses info {
                description
                    "Local PCEP entity information";
            }

            container pce-info {
                when "((../role == 'pce')" +
                     " or " +
                     "(../role == 'pcc-and-pce'))"
                {
                    description
                        "When PCEP entity is PCE";
                }
                uses pce-info {
                    description
                        "Local PCE information";
                }
                                uses authentication {
                                        description
                                                "Local PCE authentication inform
ation";
                                }
                description
                    "The Local PCE Entity PCE information";
            }

            uses pcep-entity-info{
                description
                    "The operational information related to the
                     PCEP entity.";
            }

            container stateful-parameter{
                if-feature stateful;
                must "(../info/capability/stateful/active == true)"
                {
                    error-message
                        "The Active Stateful PCE must be enabled";
                    description
                        "When PCEP entity is active stateful
                         enabled";
                }
                uses stateful-pce-parameter;

                description
                    "The operational stateful parameters";
            }
```

```
            container lsp-db{
                if-feature stateful;
                description
                    "The LSP-DB";
                list association-list {
                    key "id source global-source extended-id";
                    description
                        "List of all PCEP associations";
                    uses association {
                        description
                            "The Association attributes";
                    }
                    list lsp {
                        key "plsp-id pcc-id";
                        description
                            "List of all LSP in this association";
                        leaf plsp-id {
                            type leafref {
                                path "/pcep-state/entity/lsp-db/"
                                + "lsp/plsp-id";
                            }
                            description
                                "Reference to PLSP-ID in LSP-DB";
                        }
                        leaf pcc-id {
                            type leafref {
                                path "/pcep-state/entity/lsp-db/"
                                + "lsp/pcc-id";
                            }
                            description
                                "Reference to PCC-ID in LSP-DB";
                        }
                    }
                }
                list lsp{
                    key "plsp-id pcc-id";
                    description
                        "List of all LSPs in LSP-DB";
                    uses lsp-state{
                        description
                            "The PCEP specific attributes for
                             LSP-DB.";
                    }
                    list association-list {
                        key "id source global-source extended-id";
                        description
                            "List of all PCEP associations";
                        uses association-ref {
```

```
                        description
                            "Reference to the Association
                             attributes";
                    }
                }

            }
        }
        container peers{
            description
                    "The list of peers for the entity";

            list peer{
                key "addr";

                description
                    "The peer for the entity.";

                leaf addr {
                    type inet:ip-address;
                    description
                        "The local Internet address of this PCEP
                         peer.";
                }

                leaf role {
                    type pcep-role;
                    description
                        "The role of the PCEP Peer.
                         Takes one of the following values.
                         - unknown(0): this PCEP peer role
                           is not known.
                         - pcc(1): this PCEP peer is a PCC.
                         - pce(2): this PCEP peer is a PCE.
                         - pcc-and-pce(3): this PCEP peer
                           is both a PCC and a PCE.";
                }


                uses info {
                    description
                        "PCEP peer information";
                }


                container pce-info {
                    when "((../role == 'pce')" +
                    " or " +
```

```
            "(../role == 'pcc-and-pce'))"
            {
                description
                    "When PCEP entity is PCE";
            }
            uses pce-info {
                description
                    "PCE Peer information";
            }
        description
            "The PCE Peer information";
        }

        leaf delegation-pref{
            if-feature stateful;
            type uint8{
                range "0..7";
            }
            must "((../../role == 'pcc')" +
                 " or " +
                 "(../../role == 'pcc-and-pce'))"
            {
                error-message
                    "The PCEP entity must be PCC";
                description
                    "When PCEP entity is PCC";
            }
            must "(../../info/capability/stateful/active"
                 + " == true)"
            {
                error-message
                    "The Active Stateful PCE must be
                     enabled";
                description
                    "When PCEP entity is active stateful
                     enabled";
            }
            description
                "The PCE peer delegation preference.";
        }

        uses authentication {
            description
                "PCE Peer authentication";
        }

        leaf discontinuity-time {
            type yang:timestamp;
```

```
            description
                "The timestamp of the time when the
                 information and statistics were
                 last reset.";
        }

        leaf initiate-session {
            type boolean;
            description
                "Indicates whether the local PCEP
                 entity initiates sessions to this peer,
                 or waits for the peer to initiate a
                 session.";
        }

        leaf session-exists{
            type boolean;
            description
                "Indicates whether a session with
                 this peer currently exists.";
        }

        leaf num-sess-setup-ok{
            type yang:counter32;
            description
                "The number of PCEP sessions successfully
                 successfully established with the peer,
                 including any current session.  This
                 counter is incremented each time a
                 session with this peer is successfully
                 established.";
        }

        leaf num-sess-setup-fail{
            type yang:counter32;
            description
               "The number of PCEP sessions with the peer
                that have been attempted but failed
                before being fully established. This
                counter is incremented each time a
                session retry to this peer fails.";
        }

        leaf session-up-time{
            type yang:timestamp;
            must "(../num-sess-setup-ok != 0  or " +
                "(../num-sess-setup-ok = 0  and "  +
                "session-up-time = 0))" {
```

```
                    error-message
                        "Invalid Session Up timestamp";
                    description
                        "If num-sess-setup-ok is zero,
                         then this leaf contains zero.";
                }
            description
               "The timestamp value of the last time a
                session with this peer was successfully
                established.";
        }

        leaf session-fail-time{
            type yang:timestamp;
            must "(../num-sess-setup-fail != 0  or " +
                "(../num-sess-setup-fail = 0  and "  +
                "session-fail-time = 0))" {
                    error-message
                        "Invalid Session Fail timestamp";
                    description
                        "If num-sess-setup-fail is zero,
                         then this leaf contains zero.";
                }
            description
              "The timestamp value of the last time a
               session with this peer failed to be
               established.";
        }

        leaf session-fail-up-time{
            type yang:timestamp;
            must "(../num-sess-setup-ok != 0  or "     +
                "(../num-sess-setup-ok = 0  and "     +
                "session-fail-up-time = 0))" {
                    error-message
                        "Invalid Session Fail from
                         Up timestamp";
                    description
                        "If num-sess-setup-ok is zero,
                         then this leaf contains zero.";
                }
            description
              "The timestamp value of the last time a
               session with this peer failed from
               active.";
        }

        container pcep-stats {
```

```
                description
                    "The container for all statistics at peer
                     level.";
                uses pcep-stats{
                    description
                        "Since PCEP sessions can be
                        ephemeral, the peer statistics tracks
                        a peer even when no PCEP session
                        currently exists to that peer. The
                        statistics contained are an aggregate
                        of the statistics for all successive
                        sessions to that peer.";
                }

                leaf num-req-sent-closed{
                    type yang:counter32;
                    description
                        "The number of requests that were
                         sent to the peer and implicitly
                         cancelled when the session they were
                         sent over was closed.";
                }

                leaf  num-req-rcvd-closed{
                    type yang:counter32;
                    description
                        "The number of requests that were
                         received from the peer and
                         implicitly cancelled when the
                         session they were received over
                         was closed.";
                }
            }//pcep-stats



            container sessions {
                description
                    "This entry represents a single PCEP
                     session in which the local PCEP entity
                     participates.
                     This entry exists only if the
                     corresponding PCEP session has been
                     initialized by some event, such as
                     manual user configuration, auto-
                     discovery of a peer, or an incoming
                     TCP connection.";
```

```
                    list session {
                        key "initiator";

                        description
                            "The list of sessions, note that
                             for a time being two sessions
                             may exist for a peer";

                        leaf initiator {
                            type pcep-initiator;
                            description
                                "The initiator of the session,
                                 that is, whether the TCP
                                 connection was initiated by
                                 the local PCEP entity or the
                                 peer.
                                 There is a window during
                                 session initialization where
                                 two sessions can exist between
                                 a pair of PCEP speakers, each
                                 initiated by one of the
                                 speakers. One of these
                                 sessions is always discarded
                                 before it leaves OpenWait state.
                                 However, before it is discarded,
                                 two sessions to the given peer
                                 appear transiently in this MIB
                                 module. The sessions are
                                 distinguished by who initiated
                                 them, and so this field is the
                                 key.";
                        }

                        leaf state-last-change {
                            type yang:timestamp;
                            description
                                "The timestamp value at the
                                 time this session entered its
                                 current state as denoted by
                                 the state leaf.";
                        }

                        leaf state {
                            type pcep-sess-state;
                            description
                                "The current state of the
                                 session.
                                 The set of possible states
```

```
                excludes the idle state since
                entries do not exist in the
                idle state.";
            }

            leaf session-creation {
                type yang:timestamp;
                description
                    "The timestamp value at the
                     time this session was
                     created.";
            }

            leaf connect-retry {
                type yang:counter32;
                description
                     "The number of times that the
                      local PCEP entity has
                      attempted to establish a TCP
                      connection for this session
                      without success. The PCEP
                      entity gives up when this
                      reaches connect-max-retry.";
            }

            leaf local-id {
                type uint32 {
                  range "0..255";
                }
                description
                     "The value of the PCEP session
                      ID used by the local PCEP
                      entity in the Open message
                      for this session.
                      If state is tcp-pending then
                      this is the session ID that
                      will be used in the Open
                      message. Otherwise, this is
                      the session ID that was sent
                      in the Open message.";
            }

            leaf remote-id {
                type uint32 {
                    range "0..255";
                }
                must "((../state != 'tcp-pending'" +
                     "and " +
```

```
                    "../state != 'open-wait' )" +
                    "or " +
                    "((../state = 'tcp-pending'" +
                    " or " +
                    "../state = 'open-wait' )" +
                    "and remote-id = 0))" {
                       error-message
                           "Invalid remote-id";
                       description
                            "If state is tcp-pending
                             or open-wait then this
                             leaf is not used and
                             MUST be set to zero.";
                    }
                description
                    "The value of the PCEP session
                     ID used by the peer in its
                     Open message for this
                     session.";
            }

            leaf keepalive-timer {
                type uint32 {
                  range "0..255";
                }
                units "seconds";
                must "(../state = 'session-up'" +
                    "or " +
                    "(../state != 'session-up'" +
                    "and keepalive-timer = 0))" {
                       error-message
                           "Invalid keepalive
                            timer";
                       description
                           "This field is used if
                            and only if state is
                            session-up. Otherwise,
                            it is not used and
                            MUST be set to
                            zero.";
                    }
                description
                    "The agreed maximum interval at
                     which the local PCEP entity
                     transmits PCEP messages on this
                     PCEP session.  Zero means that
                     the local PCEP entity never
                     sends Keepalives on this
```

```
                session.";
            }

            leaf peer-keepalive-timer {
                type uint32 {
                  range "0..255";
                }
                units "seconds";
                must "(../state = 'session-up'" +
                     "or "   +
                     "(../state != 'session-up'" +
                     "and " +
                     "peer-keepalive-timer = 0))" {
                        error-message
                            "Invalid Peer keepalive
                             timer";
                        description
                            "This field is used if
                             and only if state is
                             session-up. Otherwise,
                             it is not used and MUST
                             be set to zero.";
                    }
                description
                     "The agreed maximum interval at
                      which the peer transmits PCEP
                      messages on this PCEP session.
                      Zero means that the peer never
                      sends Keepalives on this
                      session.";
            }

            leaf dead-timer {
                type uint32 {
                  range "0..255";
                }
                units "seconds";
                description
                     "The dead timer interval for
                      this PCEP session.";
            }

            leaf peer-dead-timer {
                type uint32 {
                  range "0..255";
                }
                units "seconds";
                must "((../state != 'tcp-pending'" +
```

```
                    "and " +
                    "../state != 'open-wait' )" +
                    "or " +
                    "((../state = 'tcp-pending'" +
                    " or " +
                    "../state = 'open-wait' )" +
                    "and " +
                    "peer-dead-timer = 0))" {
                       error-message
                           "Invalid Peer Dead
                            timer";
                       description
                           "If state is tcp-
                            pending or open-wait
                            then this leaf is not
                            used and MUST be set to
                            zero.";
                    }
                description
                    "The peer's dead-timer interval
                     for this PCEP session.";
            }

            leaf ka-hold-time-rem {
                type uint32 {
                  range "0..255";
                }
                units "seconds";
                must "((../state != 'tcp-pending'" +
                    "and " +
                    "../state != 'open-wait' ) " +
                    "or " +
                    "((../state = 'tcp-pending'" +
                    "or " +
                    "../state = 'open-wait' )" +
                    "and " +
                    "ka-hold-time-rem = 0))" {
                       error-message
                           "Invalid Keepalive hold
                            time remaining";
                       description
                           "If state is tcp-pending
                            or open-wait then this
                            field is not used and
                            MUST be set to zero.";
                    }
                description
                    "The keep alive hold time
```

```
                remaining for this session.";
            }

            leaf overloaded {
                type boolean;
                description
                     "If the local PCEP entity has
                      informed the peer that it is
                      currently overloaded, then this
                      is set to true.  Otherwise, it
                      is set to false.";
            }

            leaf overload-time {
                type uint32;
                units "seconds";
                must "(../overloaded = true or" +
                    "(../overloaded != true and" +
                    " overload-time = 0))" {
                      error-message
                          "Invalid overload-time";
                      description
                          "This field is only used
                           if overloaded is set to
                           true. Otherwise, it is
                           not used and MUST be set
                           to zero.";
                    }
                description
                     "The interval of time that is
                      remaining until the local PCEP
                      entity will cease to be
                      overloaded on this session.";
            }

            leaf peer-overloaded {
                type boolean;
                description
                     "If the peer has informed the
                      local PCEP entity that it is
                      currently overloaded, then this
                      is set to true. Otherwise, it
                      is set to false.";
            }

            leaf peer-overload-time {
                type uint32;
                units "seconds";
```

```
            must "(../peer-overloaded = true" +
                 " or " +
                 "(../peer-overloaded != true" +
                 " and " +
                 "peer-overload-time = 0))" {
                    error-message
                        "Invalid peer overload
                         time";
                    description
                        "This field is only used
                         if peer-overloaded is
                         set to true. Otherwise,
                         it is not used and MUST
                         be set to zero.";
                }
            description
                 "The interval of time that is
                  remaining until the peer will
                  cease to be overloaded.  If it
                  is not known how long the peer
                  will stay in overloaded state,
                  this leaf is set to zero.";
        }
        leaf lspdb-sync {
            if-feature stateful;
            type sync-state;
            description
                "The LSP-DB state synchronization
                status.";
        }
        leaf discontinuity-time {
            type yang:timestamp;
            description
                 "The timestamp value of the time
                  when the statistics were last
                  reset.";
        }

        container pcep-stats {
            description
                "The container for all statistics
                 at session level.";
            uses pcep-stats{
                description
                    "The statistics contained are
                     for the current sessions to
                     that peer. These are lost
                     when the session goes down.
```

```
                                        ";
                                    }
                                }//pcep-stats

                            } // session
                        } // sessions
                    }//peer
                }//peers
            }//entity
        }//pcep-state

        /*
         * Notifications
         */
        notification pcep-session-up {
            description
                "This notification is sent when the value of
                 '/pcep/pcep-state/peers/peer/sessions/session/state'
                 enters the 'session-up' state.";

            uses notification-instance-hdr;

            uses notification-session-hdr;

            leaf state-last-change {
                type yang:timestamp;
                description
                    "The timestamp value at the time this session entered
                    its current state as denoted by the state leaf.";
            }

            leaf state {
                type pcep-sess-state;
                description
                    "The current state of the session.
                     The set of possible states excludes the idle state
                     since entries do not exist in the idle state.";
            }
        } //notification

        notification pcep-session-down {
            description
                "This notification is sent when the value of
                 '/pcep/pcep-state/peers/peer/sessions/session/state'
                 leaves the 'session-up' state.";

            uses notification-instance-hdr;
```

```
        leaf session-initiator {
            type pcep-initiator;
            description
                "The initiator of the session.";
        }

        leaf state-last-change {
            type yang:timestamp;
            description
                "The timestamp value at the time this session entered
                its current state as denoted by the state leaf.";
        }

        leaf state {
            type pcep-sess-state;
            description
                "The current state of the session.
                 The set of possible states excludes the idle state
                 since entries do not exist in the idle state.";
        }
    } //notification

    notification pcep-session-local-overload {
        description
            "This notification is sent when the local PCEP entity
            enters overload state for a peer.";

        uses notification-instance-hdr;

        uses notification-session-hdr;

        leaf overloaded {
            type boolean;
            description
                 "If the local PCEP entity has informed the peer that
                 it is currently overloaded, then this is set to
                 true. Otherwise, it is set to false.";
        }

        leaf overload-time {
            type uint32;
            units "seconds";
            must "(../overloaded = true or "    +
                "(../overloaded != true  and "  +
                "overload-time = 0))" {
                    error-message
                        "Invalid overload-time";
                    description
```

```
                        "This field is only used if overloaded is
                         set to true. Otherwise, it is not used
                         and MUST be set to zero.";
                    }
            description
                 "The interval of time that is remaining until the
                 local PCEP entity will cease to be overloaded on
                 this session.";
        }
    } //notification

    notification pcep-session-local-overload-clear {
        description
            "This notification is sent when the local PCEP entity
            leaves overload state for a peer.";

        uses notification-instance-hdr;

        leaf overloaded {
            type boolean;
            description
                 "If the local PCEP entity has informed the peer
                 that it is currently overloaded, then this is set
                 to true.  Otherwise, it is set to false.";
        }
    } //notification

    notification pcep-session-peer-overload {
        description
            "This notification is sent when a peer enters overload
            state.";

        uses notification-instance-hdr;

        uses notification-session-hdr;

        leaf peer-overloaded {
            type boolean;
            description
                "If the peer has informed the local PCEP entity that
                it is currently overloaded, then this is set to true.
                Otherwise, it is set to false.";
        }

        leaf peer-overload-time {
            type uint32;
            units "seconds";
            must "(../peer-overloaded = true or "    +
```

```
                "(../peer-overloaded != true  and "  +
                "peer-overload-time = 0))" {
                    error-message
                        "Invalid peer-overload-time";
                    description
                        "This field is only used if
                         peer-overloaded is set to true.
                         Otherwise, it is not used and MUST
                         be set to zero.";
                }
            description
                "The interval of time that is remaining until the
                peer will cease to be overloaded.  If it is not known
                how long the peer will stay in overloaded state, this
                leaf is set to zero.";
        }
    } //notification

    notification pcep-session-peer-overload-clear {
        description
            "This notification is sent when a peer leaves overload
            state.";

        uses notification-instance-hdr;

        leaf peer-overloaded {
            type boolean;
            description
                "If the peer has informed the local PCEP entity that
                it is currently overloaded, then this is set to true.
                Otherwise, it is set to false.";
        }
    } //notification
}//module

<CODE ENDS>
```

9. 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 the 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.

TBD: List specific Subtrees and data nodes and their sensitivity/ vulnerability.

10. Manageability Considerations

10.1. Control of Function and Policy

10.2. Information and Data Models

10.3. Liveness Detection and Monitoring

10.4. Verify Correct Operations

10.5. Requirements On Other Protocols

10.6. Impact On Network Operations

11. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

URI: urn:ietf:params:xml:ns:yang:ietf-pcep

Registrant Contact: The PCE WG of the IETF.

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-pcep
Namespace:    urn:ietf:params:xml:ns:yang:ietf-pcep
Prefix:       pcep
Reference:    This I-D
```

12. Acknowledgements

The initial document is based on the PCEP MIB [RFC7420]. Further this document structure is based on Routing Yang Module [I-D.ietf-netmod-routing-cfg]. We would like to thank the authors of aforementioned documents.

13. References

13.1. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <http://www.rfc-editor.org/info/rfc5440>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[I-D.ietf-pce-stateful-pce] Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-14 (work in progress), March 2016.

[I-D.ietf-pce-pce-initiated-lsp] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model", draft-ietf-pce-pce-initiated-lsp-05 (work in progress), October 2015.

[I-D.ietf-pce-lsp-setup-type]
Sivabalan, S., Medved, J., Minei, I., Crabbe, E., Varga, R., Tantsura, J., and J. Hardwick, "Conveying path setup type in PCEP messages", draft-ietf-pce-lsp-setup-type-03 (work in progress), June 2015.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Lopez, V., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-07 (work in progress), March 2016.

13.2. Informative References

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <http://www.rfc-editor.org/info/rfc4655>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7420] Koushik, A., Stephan, E., Zhao, Q., King, D., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Management Information Base (MIB) Module", RFC 7420, DOI 10.17487/RFC7420, December 2014, <http://www.rfc-editor.org/info/rfc7420>.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-06 (work in progress), March 2016.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-03 (work in progress), March 2016.

Appendix A. Contributor Addresses

Rohit Pobbathi
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: rohit.pobbathi@huawei.com

Vinod KumarS
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: vinods.kumar@huawei.com

Zafar Ali
Cisco Systems
Canada

EMail: zali@cisco.com

Xufeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna, VA 22182
USA

EMail: xufeng.liu@ericsson.com

Young Lee
Huawei Technologies
5340 Legacy Drive, Building 3
Plano, TX 75023, USA

Phone: (469) 277-5838
EMail: leeyoung@huawei.com

Udayasree Palle
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: udayasree.palle@huawei.com

Xian Zhang
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R.China

EMail: zhang.xian@huawei.com

Avantika
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: avantika.sushilkumar@huawei.com

Authors' Addresses

Dhruv Dhody (editor)
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: dhruv.ietf@gmail.com

Jonathan Hardwick
Metaswitch
100 Church Street
Enfield EN2 6BQ
UK

EMail: jonathan.hardwick@metaswitch.com

Vishnu Pavan Beeram
Juniper Networks
USA

EMail: vbeeram@juniper.net

Jeff Tantsura
USA

EMail: jefftant@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2018

Yimin Shen
Minto Jeyananth
Juniper Networks
Bruno Decraene
Orange
Hannes Gredler
RtBrick Inc
Carsten Michel
Deutsche Telekom
Huaimo Chen
Yuanlong Jiang
Huawei Technologies Co., Ltd.
November 14, 2017

MPLS Egress Protection Framework
draft-shen-mpls-egress-protection-framework-07

Abstract

This document specifies a fast reroute framework for protecting IP/MPLS services and MPLS transport tunnels against egress node and egress link failures. In this framework, the penultimate-hop router of an MPLS tunnel acts as the point of local repair (PLR) for egress node failure, and the egress router of the MPLS tunnel acts as the PLR for egress link failure. Each of them pre-establishes a bypass tunnel to a protector. Upon an egress node or link failure, the corresponding PLR performs local failure detection and local repair, by rerouting packets over the corresponding bypass tunnel. The protector in turn performs context label switching or context IP forwarding to send the packets to the ultimate service destination(s). This mechanism can be used to reduce traffic loss before global repair reacts to the failure and control plane protocols converge on the topology changes due to the failure. The framework is applicable to all types of IP/MPLS services and MPLS tunnels. Under the framework, service protocol extensions may be further specified to support service label distribution to the protector.

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction 3
2. Specification of Requirements 5
3. Terminology 5
4. Requirements 7
5. Egress node protection 8
  5.1. Reference topology 8
  5.2. Egress node failure and detection 8
  5.3. Protector and PLR 9
  5.4. Protected egress 10
  5.5. Egress-protected tunnel and service 11
  5.6. Egress-protection bypass tunnel 11
  5.7. Context ID, context label, and context based forwarding . 12
  5.8. Advertisement and path resolution for context ID 14
  5.9. Egress-protection bypass tunnel establishment 15
  5.10. Local repair on PLR 15
  5.11. Service label distribution from egress router to protector 16
  5.12. Centralized protector mode 16
6. Egress link protection 18
7. Global repair 21
8. Example: Layer-3 VPN egress protection 21
  8.1. Egress node protection 23
  8.2. Egress link protection 24
  8.3. Global repair 24

9. IANA Considerations 24
10. Security Considerations 24
11. Acknowledgements 25
12. References 25
12.1. Normative References 25
12.2. Informative References 25
Authors' Addresses 26

1. Introduction

In MPLS networks, label switched paths (LSPs) are widely used as transport tunnels to carry IP and MPLS services across MPLS domains. Examples of MPLS services are layer-2 VPNs, layer-3 VPNs, hierarchical LSPs, and others. In general, a tunnel may carry multiple services of one or multiple types, if the tunnel can satisfy both individual and aggregate requirements (e.g. CoS, QoS) of these services. The egress router of the tunnel should host the corresponding service instances of the services. An MPLS service instance is responsible for forwarding service packets via an egress link to the service destination, based on a service label. An IP service instance is responsible for doing the same based on a service IP address. The egress link is often called a PE-CE (provider edge - customer edge) link or attachment circuit (AC).

Today, local repair based fast reroute mechanisms [RFC4090], [RFC5286], [RFC7490], [RFC7812] have been widely deployed to protect MPLS tunnels against transit link/node failures. They can achieve fast restoration of traffic in the order of tens of milliseconds. Local repair refers to the scenario where the router upstream to an anticipated failure (aka. PLR, i.e. point of local repair) pre-establishes a bypass tunnel to the router downstream of the failure (aka. MP, i.e. merge point), and pre-installs the forwarding state of the bypass tunnel in the data plane. The PLR also uses a rapid mechanism (e.g. link layer OAM, BFD, and others) to locally detect the failure in the data plane. When the failure occurs, the PLR reroutes traffic through the bypass tunnel to the MP, allowing the traffic to continue to flow to the tunnel's egress router.

This document describes a fast reroute framework for egress node and egress link protection. Similar to transit link/node protection, this framework relies on a PLR to perform local failure detection and local repair. In egress node protection, the PLR is the penultimate-hop router of a tunnel. In egress link protection, the PLR is the egress router of the tunnel. The framework relies on a so-called "protector" to serve as the tailend of a bypass tunnel. The protector is a router that hosts "protection service instances" and has its own connectivity or paths to service destinations. When a PLR is doing local repair, the protector is responsible for

performing "context label switching" for rerouted MPLS service packets and "context IP forwarding" for rerouted IP service packets. Thus, the service packets can continue to reach service destinations with minimum disruption.

This framework considers an egress node failure as a failure of a tunnel, as well as a failure of all the services carried by the tunnel, because service packets can no longer reach the service instances on the egress router. Therefore, the framework addresses egress node protection at both tunnel level and service level simultaneously. Likewise, the framework considers an egress link failure as a failure of all the services traversing the link, and addresses egress link protection at the service level.

This framework requires that the destination (a CE or site) of a service MUST be dual-homed or have dual paths to an MPLS network, normally via two MPLS edge routers. One of them is the egress router of the service's transport tunnel, and the other is a backup egress router which hosts "backup service instances". In the "co-located" protector mode in this document, the backup egress router serves as a protector, and hence each backup service instance acts as a protection instance. In the "centralized" protector mode (Section 5.12), a protector and a backup egress router are decoupled, and each protection service instance and its corresponding backup service instance are hosted on separate routers.

The framework is described by mainly referring to P2P (point-to-point) tunnels. However, it is equally applicable to P2MP (point-to-multipoint), MP2P (multipoint-to-point) and MP2MP (multipoint-to-multipoint) tunnels, when a sub-LSP can be viewed as a P2P tunnel.

The framework is a multi-service and multi-transport framework. It assumes a generic model where each service is comprised of a common set of components, including a service instance, a service label, and a service label distribution protocol, and the service is transported over an MPLS tunnel of any type. The framework also assumes service labels to be downstream assigned, i.e. assigned by egress routers. Therefore, the framework is generally applicable to most existing and future services. Services which use upstream-assigned service labels are out of scope of this document and left for further study.

The framework does not require extensions for the existing signaling and label distribution protocols (e.g. RSVP, LDP, BGP, etc.) of MPLS tunnels. It expects transport tunnels and bypass tunnels to be established by using the generic mechanisms provided by the protocols. On the other hand, it does not preclude future extensions to the protocols which may facilitate the procedures. One example of such extension is [RSVP-EP]. The framework may need extensions for

IGPs and service label distribution protocols, to support protection establishment and context label switching. This document provides guidelines for these extensions, but the specific details SHOULD be addressed in separate documents.

The framework is intended to complement control-plane convergence and global repair, which are traditionally used to recover networks from egress node and egress link failures. Control-plane convergence relies on control protocols to react on the topology changes due to a failure. Global repair relies an ingress router to remotely detect a failure and switch traffic to an alternative path. An example of global repair is the BGP Prefix Independent Convergence mechanism [BGP-PIC] for BGP established services. Compared with these mechanisms, this framework is considered as faster in traffic restoration, due to the nature of local failure detection and local repair. However, it is RECOMMENDED that the framework SHOULD be used in conjunction with control-plane convergence or global repair, in order to take the advantages of both approaches to achieve more effective protection. That is, the framework provides fast and temporary repair, and control-plane convergence or global repair provides ultimate and permanent repair.

2. Specification of Requirements

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

3. Terminology

Egress router - A router at the egress endpoint of a tunnel. It hosts service instances for all the services carried by the tunnel, and has connectivity with the destinations of the services.

Egress node failure - A failure of an egress router.

Egress link failure - A failure of the egress link (e.g. PE-CE link, attachment circuit) of a service.

Egress failure - An egress node failure or an egress link failure.

Egress-protected tunnel - A tunnel whose egress router is protected by a mechanism according to this framework. The egress router is hence called a protected egress router.

Egress-protected service - An IP or MPLS service which is carried by an egress-protected tunnel, and hence protected by a mechanism according to this framework.

Backup egress router - Given an egress-protected tunnel and its egress router, this is another router which has connectivity with all or a subset of the destinations of the egress-protected services carried by the egress-protected tunnel.

Backup service instance - A service instance which is hosted by a backup egress router, and corresponding to an egress-protected service on a protected egress router.

Protector - A role acted by a router as an alternate of a protected egress router, to handle service packets in the event of an egress failure. A protector may be physically co-located with or decoupled from a backup egress router, depending on the co-located or centralized protector mode.

Protection service instance - A service instance hosted by a protector, corresponding to the service instance of an egress-protected service on a protected egress router. A protection service instance is a backup service instance, if the protector is co-located with a backup egress router.

PLR - A router at the point of local repair. In egress node protection, it is the penultimate-hop router on an egress-protected tunnel. In egress link protection, it is the egress router of the egress-protected tunnel.

Protected egress {E, P} - A virtual node consisting of an ordered pair of egress router E and protector P. It serves as the virtual destination of an egress-protected tunnel, and as the virtual location of the egress-protected services carried by the tunnel.

Context identifier (ID) - A globally unique IP address assigned to a protected egress {E, P}.

Context label - A non-reserved label assigned to a context ID by a protector.

Egress-protection bypass tunnel - A tunnel used to reroute service packets around an egress failure.

Co-located protector mode - The scenario where a protector and a backup egress router are co-located as one router, and hence each backup service instance serves as a protection service instance.

Centralized protector mode - The scenario where a protector is a dedicated router, and is decoupled from backup egress routers.

Context label switching - Label switching performed by a protector, in the label space of an egress router indicated by a context label.

Context IP forwarding - IP forwarding performed by a protector, in the IP address space of an egress router indicated by a context label.

4. Requirements

This document considers the followings as the design requirements of this egress protection framework.

- The framework must support P2P tunnels. It should equally support P2MP, MP2P and MP2MP tunnels, by treating each sub-LSP as a P2P tunnel.

- The framework must support multi-service and multi-transport networks. It must accommodate existing and future signaling and label-distribution protocols of tunnels and bypass tunnels, including RSVP, LDP, BGP, IGP, segment routing, and others. It must also accommodate existing and future IP/MPLS services, including layer-2 VPNs, layer-3 VPNs, hierarchical LSP, and others. It must provide a generic solution for environments where different types of services and tunnels may co-exist.

- The framework must consider minimizing disruption during deployment. It should only involve routers close to egress, and be transparent to ingress routers and other transit routers.

- In egress node protection, for scalability and performance reasons, a PLR must be agnostic to services and service labels, like PLRs in transit link/node protection. It must maintain bypass tunnels and bypass forwarding state on a per-transport-tunnel basis, rather than per-service-destination or per-service-label basis. It should also support bypass tunnel sharing between transport tunnels.

- A PLR must be able to use its local visibility or information of routing and/or TE topology to compute or resolve a path for a bypass tunnel to a protector.

- A protector must be able to perform context label switching for rerouted MPLS service packets, based on service label(s) assigned by an egress router. It must be able to perform context IP forwarding for rerouted IP service packets, in the public or private IP address space used by an egress router.

- The framework must be able to work seamlessly with transit link/ node protection mechanisms to achieve end-to-end coverage.

- The framework must be able to work in conjunction with global repair and control plane convergence.

5. Egress node protection

5.1. Reference topology

This document refers to the following topology when describing the procedures of egress node protection.

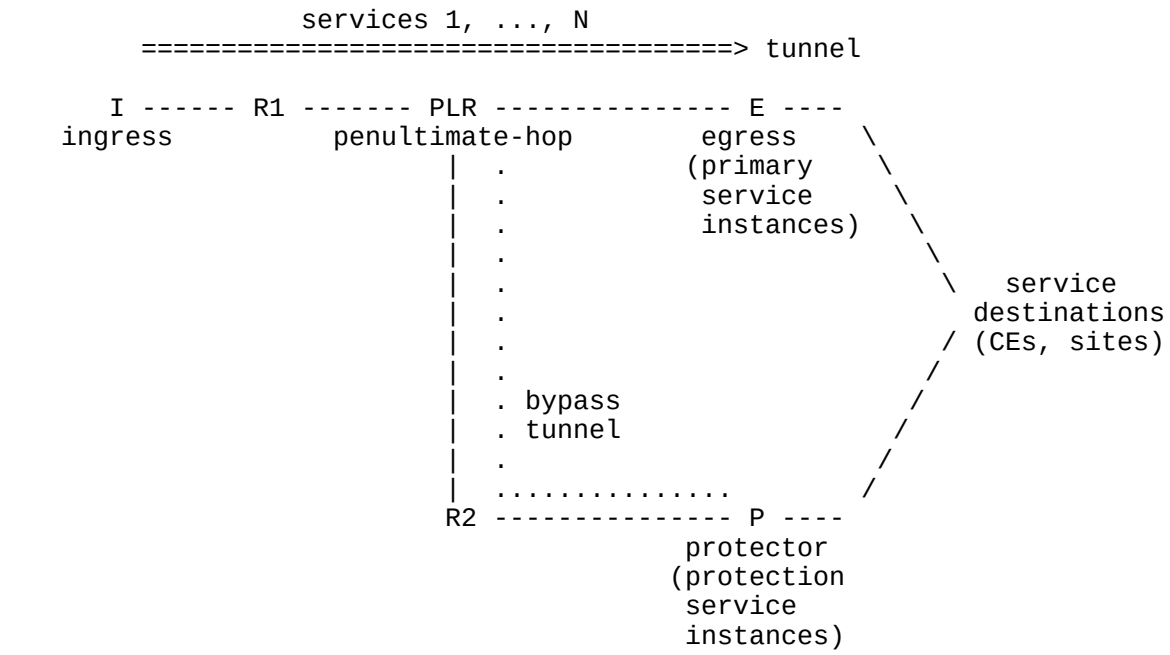


Figure 1

5.2. Egress node failure and detection

An egress node failure refers to the failure of an MPLS tunnel's egress router. At the service level, it also means a service instance failure for each IP/MPLS service carried by the tunnel.

Ideally, an egress node failure can be detected by an adjacent router (i.e. PLR in this framework) using a node liveness detection

mechanism, or based on a collective failure of all the links to that node. However, the assumption is that the mechanisms SHOULD be reasonably fast, i.e. faster than control plane failure detection and remote failure detection. Otherwise, local repair will not be able to provide much benefit compared to control plane convergence or global repair. In general, the speed, accuracy, and reliability of a mechanism are the key factors to decide its applicability in egress node protection. This document provides the following guidelines in this regard.

- If the PLR has a reasonably fast mechanism to detect and differentiate a link failure (of the link between the PLR and the egress node) and an egress node failure, it SHOULD set up both link protection and egress node protection, and trigger one and only one protection upon a corresponding failure.

- If the PLR has a fast mechanism to detect a link failure and an egress node failure, but cannot distinguish them; Or, if the PLR has a fast mechanism to detect a link failure only, but not an egress node failure, the PLR has two options:

  1. It MAY set up link protection only, and leave the egress node failure to global repair and control plane convergence to handle.

  2. It MAY set up egress node protection only, and treat a link failure as a trigger for the egress node protection. However, the assumption is that treating a link failure as an egress node failure MUST NOT have a negative impact on services. Otherwise, it SHOULD adopt the previous option.

5.3. Protector and PLR

A router is assigned to the "protector" role to protect a tunnel and the services carried by the tunnel against an egress node failure. The protector is responsible for hosting a protection service instance for each protected service, serving as the tailend of a bypass tunnel, and performing context label switching and/or context IP forwarding for rerouted service packets.

A tunnel can be protected by only one protector at a given time. Multiple tunnels to a given egress router may be protected by a common protector or different protectors. A protector may protect multiple tunnels with a common egress router or different egress routers.

For each tunnel, its penultimate-hop router acts as a PLR. The PLR pre-establishes a bypass tunnel to the protector, and pre-installs

bypass forwarding state in the data plane. Upon detection of an egress node failure, the PLR reroutes all the service packets received on the tunnel though the bypass tunnel to the protector. For MPLS service packets, the PLR keeps service labels intact in the packets. The protector in turn forwards the rerouted service packets towards the ultimate service destinations. Specifically, it performs context label switching for MPLS service packets, based on service labels assigned by the protected egress router; It performs context IP forwarding for IP service packets, based on their destination addresses.

The protector MUST have its own connectivity with each service destination, via a direct link or a multi-hop path, which MUST NOT traverse the protected egress router or be affected by the egress node failure. This also requires that each service destination MUST be dual-homed or have dual paths to the egress router and a backup egress router which serves as the protector. Each protection service instance on the protector relies on such connectivity to set up forwarding state for context label switching and/or context IP forwarding.

5.4. Protected egress

This document introduces the notion of "protected egress" as a virtual node consisting of the egress router E of a tunnel and a protector P. It is denoted by an ordered pair of {E, P}, indicating the primary-and-protector relationship between the two routers. It serves as the virtual destination of the tunnel, and the virtual location of service instances for the services carried by the tunnel. The tunnel and services are considered as being "associated" with the protected egress {E, P}.

A given egress router E may be the tailend of multiple tunnels. In general, the tunnels may be protected by multiple protectors, e.g. P1, P2, and so on, with each Pi protecting a subset of the tunnels. Thus, these routers form multiple protected egresses, i.e. {E, P1} , {E, P2}, and so on. Each tunnel is associated with one and only one protected egress {E, Pi}. All the services carried by the tunnel are then automatically associated with the same protected egress {E, Pi}. Conversely, a service associated with a protected egress {E, Pi} MUST be carried by a tunnel associated with the protected egress {E, Pi}. This mapping MUST be ensured by the ingress router of the tunnel and the service (Section 5.5).

Two routers X and Y may be protectors for each other. In this case, they form two distinct protected egresses {X, Y} and {Y, X}.

5.5. Egress-protected tunnel and service

A tunnel, which is associated with a protected egress {E, P}, is called an egress-protected tunnel. It is associated with one and only one protected egress {E, P}. Multiple egress-protected tunnels may be associated with a given protected egress {E, P}. In this case, they share the common egress router and protector, but may or may not share a common ingress router, or a common PLR (i.e. penultimate-hop router).

An egress-protected tunnel is considered as logically "destined" for its protected egress {E, P}. However, its path MUST be resolved and established with E as the physical tailend.

A service, which is associated with a protected egress {E, P}, is called an egress-protected service. The egress router E hosts the primary instance of the service, and the protector P hosts the protection instance of the service.

An egress-protected service is associated with one and only one protected egress {E, P}. Multiple egress-protected services may be associated with a given protected egress {E, P}. In this case, these services share the common egress router and protector, but may or may not share a common egress-protected tunnel or a common ingress router.

An egress-protected service MUST be mapped to an egress-protected tunnel by its ingress router, based on the common protected egress {E, P} of the service and the tunnel. This is achieved by introducing the notion of "context ID" for protected egress {E, P}, as described in (Section 5.7).

5.6. Egress-protection bypass tunnel

An egress-protected tunnel destined for a protected egress {E, P} MUST have a bypass tunnel from its PLR to the protector P. This bypass tunnel is called an egress-protection bypass tunnel. The bypass tunnel is considered as logically "destined" for the protected egress {E, P}. However, due to its bypass nature, it MUST be resolved and established with P as the physical tailend and E as the node to avoid. The bypass tunnel MUST have the property that it MUST NOT be affected by any topology change caused by an egress node failure.

An egress-protection bypass tunnel is associated with one and only one protected egress {E, P}. A PLR may share an egress-protection bypass tunnel for multiple egress-protected tunnels associated with a common protected egress {E, P}. For multiple egress-protected tunnels associated with a common protected egress {E, P}, there may be one or

multiple egress-protection bypass tunnels from one or multiple PLRs to the protector P, depending on the paths of the egress-protected tunnels.

5.7. Context ID, context label, and context based forwarding

In this framework, a globally unique IPv4/v6 address is assigned to a protected egress {E, P} to serve as the identifier of the protected egress {E, P}. It is called a "context ID" due to its specific usage in context label switching and context IP forwarding on the protector. It is an IP address that is logically owned by both the egress router and the protector. For the egress node, it indicates the protector. For the protector, it indicates the egress router, particularly the egress router's forwarding context. For other routers in the network, it is an address reachable via both the egress router and the protector in the routing domain and the TE domain (Section 5.8), similar to an anycast address.

The main purpose of a context ID is to coordinate ingress router, egress router, PLR and protector in setting up egress protection. Given an egress-protected service associated with a protected egress {E, P}, its context ID is used as below:

- If the service is an MPLS service, when E distributes a service label binding message to the ingress router, E attaches the context ID to the message. If the service is an IP service, when E advertises the service destination address to the ingress router, E also attaches the context ID to the advertisement message. How the context ID is encoded in the messages is a choice of the service protocol, and may need protocol extensions to define a "context ID" object.

- The ingress router uses the context ID as destination to establish or resolve an egress-protected tunnel. The ingress router then maps the service to the tunnel for transportation. In this process, the special semantics of the context ID is transparent to the ingress router. The ingress router only treats the context ID as an IP address of E, and behaves in the same manner as in establishing or resolving a regular transport tunnel, although the end result is an egress-protected tunnel.

- The context ID is conveyed to the PLR by the signaling protocol of the egress-protected tunnel, or learned by the PLR via an IGP (i.e. OSPF or ISIS) or a topology-driven label distribution protocol (e.g. LDP). The PLR uses the context ID as destination to establish or resolve an egress-protection bypass tunnel to P while avoiding E.

- P maintains a dedicated label space or a dedicated IP address space for E, depending on whether the service is MPLS or IP. This is referred to as "E's label space" or "E's IP address space", respectively. P uses the context ID to identify the space.

- If the service is an MPLS service, E also distributes the service label binding message to P. This is the same label binding message that E advertises to the ingress router, attached with the context ID. Based on the context ID, P installs the service label in an MPLS forwarding table corresponding to E's label space. If the service is an IP service, P installs an IP route in an IP forwarding table corresponding to E's IP address space. In either case, the protection service instance on P interprets the service and constructs forwarding state for the route based on P's own connectivity to the service's destination.

- P assigns a non-reserved label to the context ID. In the data plane, this label represents the context ID and indicates E's label space and IP address space. Therefore, it is called a "context label".

- The PLR may establish the egress-protection bypass tunnel to P in several manners. If the bypass tunnel is established by RSVP, the PLR signals the bypass tunnel with the context ID as destination, and P binds the context label to the bypass tunnel. If the bypass tunnel is established by LDP, P advertises the context label for the context ID as an IP prefix FEC. If the bypass tunnel is established by the PLR in a hierarchical manner, the PLR treats the context label as a one-hop LSP over a regular bypass tunnel to P (e.g. a bypass tunnel to P's loopback IP address). If the bypass tunnel is constructed by using segment routing, the bypass tunnel is represented by a stack of SID labels with the context label as the inner-most SID label (Section 5.9). In any case, the bypass tunnel is a UHP tunnel whose incoming label at P is the context label.

- During local repair, all the service packets received by P on the bypass tunnel have the context label as top label. P first pops the context label. For an MPLS service packet, P further looks up the service label in E's label space indicated by the context label, which is called context label switching. For an IP service packet, P looks up the IP destination address in E's IP address space indicated by the context label, which is called context IP forwarding.

5.8. Advertisement and path resolution for context ID

Path resolution are computation for a context ID are done on ingress routers for egress-protected tunnels, and on PLRs for egress-protection bypass tunnels. Therefore, given a protected egress {E, P} and its context ID, E and P MUST coordinate the context ID in the routing domain and the TE domain via IGP advertisement. The context ID MUST be advertised in such a manner that all egress-protected tunnels MUST have E as tailend, and all egress-protection bypass tunnels MUST have P as tailend while avoiding E.

This document suggests two approaches:

1. The first approach is called "proxy mode". It requires E and P, but not the PLR, to have the knowledge of the egress protection schema. E and P advertise the context ID as a virtual proxy node (i.e. a logical node) connected to the two routers, with the link between the proxy node and E having more preferable IGP and TE metrics than the link between the proxy node and P. Therefore, all egress-protected tunnels destined for the context ID should automatically follow the shortest IGP or TE paths to E. Each PLR will no longer view itself as a penultimate-hop, but rather two hops away from the proxy node, via E. The PLR will be able to find a bypass path via P to the proxy node, while the bypass tunnel should actually be terminated by P.

2. The second approach is called "alias mode". It requires P and the PLR, but not E, to have the knowledge of the egress protection schema. E simply advertises the context ID as a regular IP address. P advertises the context ID and the context label by using a "context ID label binding" advertisement. The advertisement MUST be understood by the PLR. In both routing domain and TE domain, the context ID is only reachable via E. This ensures that all egress-protected tunnels destined for the context ID should have E as tailend. Based on the "context ID label binding" advertisement, the PLR can establish an egress-protection bypass tunnel in several manners (Section 5.9). The "context ID label binding" advertisement is defined as IGP mirroring context segment in [SR-ARCH], [SR-OSPF] and [SR-ISIS]. These IGP extensions are generic in nature, and hence can be used for egress protection purposes.

In a scenario where an egress-protected tunnel is an inter-area or inter-AS tunnel, its associated context ID MUST be propagated from the residing area/AS to the other areas/AS' via IGP or BGP, so that the ingress router of the tunnel can have the reachability to the context ID. The propagation process of the context ID SHOULD be the same as that of a regular IP address in an inter-area/AS environment.

5.9. Egress-protection bypass tunnel establishment

A PLR MUST know the context ID of a protected egress {E, P} in order to establish an egress-protection bypass tunnel. The information is obtained from the signaling or label distribution protocol of the egress-protected tunnel. The PLR may or may not need to have the knowledge of the egress protection schema. All it does is to set up a bypass tunnel to a context ID while avoiding the next-hop router (i.e. egress router). This is achievable by using a constraint-based computation algorithm similar to those which are commonly used in the computation of traffic engineering paths and loop-free alternate (LFA) paths. Since the context ID is advertised in the routing domain and the TE domain by IGP according to Section 5.8, the PLR should be able to resolve or establish such a bypass path with the protector as tailend. In some cases like the proxy mode, the PLR may do so in the same manner as transit node protection.

An egress-protection bypass tunnel may be established via several methods:

(1) It may be established by a signaling protocol (e.g. RSVP), with the context ID as destination. The protector binds the context label to the bypass tunnel.

(2) It may be formed by a topology driven protocol (e.g. LDP with various LFA mechanisms). The protector advertises the context ID as an IP prefix FEC, with the context label bound to it.

(3) It may be constructed as a hierarchical tunnel. When the protector uses the alias mode (Section 5.8), the PLR will have the knowledge of the context ID, context label, and protector (i.e. the advertiser). The PLR can then establish the bypass tunnel in a hierarchical manner, with the context label as a one-hop LSP over a regular bypass tunnel to the protector’s IP address (e.g. loopback address). This regular bypass tunnel may be established by RSVP, LDP, segment routing, and others.

5.10. Local repair on PLR

In this framework, a PLR is agnostic to services and service labels. This obviates the need to maintain bypass forwarding state on a per-service basis, and allows bypass tunnel sharing between egress-protected tunnels. The PLR may share an egress-protection bypass tunnel for multiple egress-protected tunnels associated with a common protected egress {E, P}. During local repair, the PLR reroutes all service packets received on the egress-protected tunnels via the egress-protection bypass tunnel. Service labels remain intact in MPLS service packets.

Label operation during the rerouting depends on the bypass tunnel's characteristics. If the bypass tunnel is a single level tunnel, the rerouting will involve swapping the incoming label of an egress-protected tunnel to the outgoing label of the bypass tunnel. If the bypass tunnel is a hierarchical tunnel, the rerouting will involve swapping the incoming label of an egress-protected tunnel to a context label, and pushing the outgoing label of a regular bypass tunnel. If the bypass tunnel is constructed by segment routing, the rerouting will involve swapping the incoming label of an egress-protected tunnel to a context label, and pushing a stack of SID labels of the bypass tunnel.

5.11. Service label distribution from egress router to protector

As mentioned in previous sections, when a protector receives a rerouted MPLS service packet, it performs context label switching based on the packet's service label which is assigned by the corresponding egress router. In order to achieve this, the protector MUST maintain such kind of service labels in dedicated label spaces on a per protected egress {E, P} basis, i.e. one label space for each egress router that it protects.

Also, there MUST be a service label distribution protocol session between each egress router and the protector. Through this protocol, the protector learns the label binding of each egress-protected service. This is the same label binding that the egress router advertises to the corresponding ingress router, attached with a context ID. The corresponding protection service instance on the protector recognizes the service, and resolves forwarding state based on its own connectivity with the service's destination. It then installs the service label with the forwarding state in the label space of the egress router, which is indicated by the context ID (i.e. context label).

Different service protocols may use different mechanisms for such kind of label distribution. Specific protocol extensions may be needed on a per-protocol basis or per-service-type basis. The details of the extensions SHOULD be specified in separate documents. As an example, RFC 8104 specifies the LDP extensions for pseudowire services.

5.12. Centralized protector mode

In this framework, it is assumed that the service destination of an egress-protected service MUST be dual-homed to two edge routers of an MPLS network. One of them is the protected egress router, and the other is a backup egress router. So far in this document, the discussion has been focusing on the scenario where a protector and a

backup egress router are co-located as one router. Therefore, the number of protectors in a network is equal to the number of backup egress routers. As another scenario, a network may assign a small number of routers to serve as dedicated protectors, each protecting a subset of egress routers. These protectors are called centralized protectors.

Topologically, a centralized protector may be decoupled from all backup egress routers, or it may be co-located with one backup egress router while decoupled from the other backup egress routers. The procedures in this section assume the scenario where a protector and a backup egress router are decoupled.

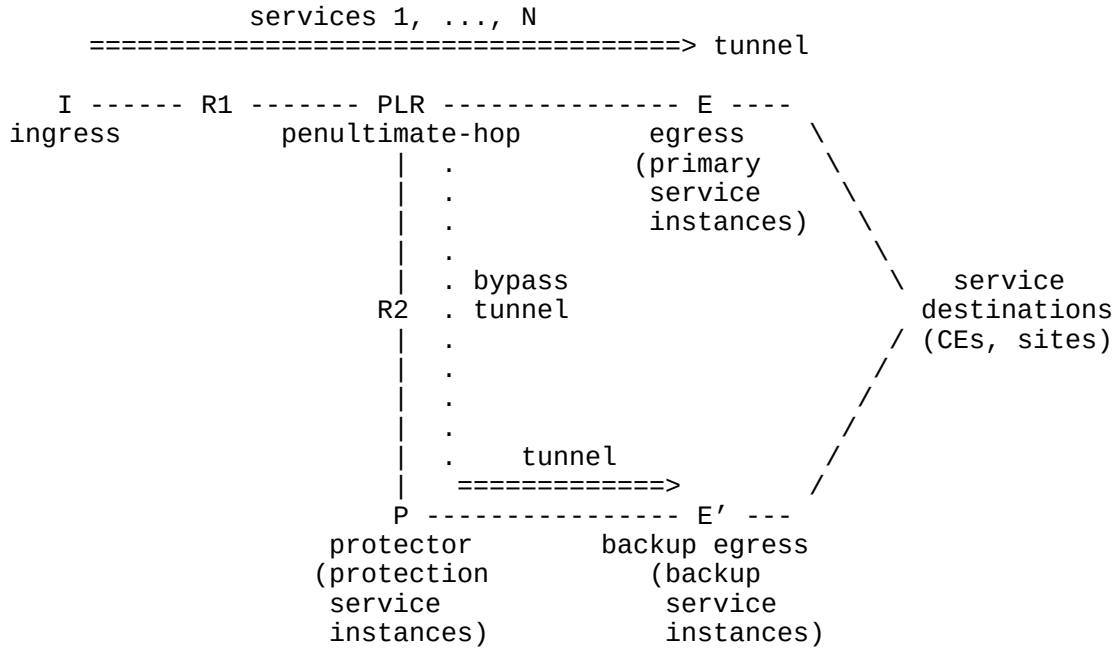


Figure 2

Like a co-located protector, a centralized protector hosts protection service instances, receives rerouted service packets from PLRs, and performs context label switching and/or context IP forwarding. For each service, instead of sending service packets directly to the service destination, the protector MUST send them via another transport tunnel to the corresponding backup service instance on a backup egress router. The backup service instance in turn forwards

them to the service destination. Specifically, in the case of an MPLS service, the protector MUST swap the service label in each received service packet to the label of the backup service advertised by the backup egress router, and then push the label (or label stack) of the transport tunnel.

In order for a centralized protector to map an egress-protected MPLS service to a service hosted on a backup egress router, there MUST be a service label distribution protocol session between the backup egress router and the protector. Through this session, the backup egress router advertises the service label of the backup service, attached with the FEC of the egress-protected service and the context ID of the protected egress {E, P}. Based on this information, the protector associates the egress-protected service with the backup service, resolves or establishes a transport tunnel to the backup egress router, and accordingly sets up forwarding state for the label of the egress-protected service in the label space of the egress router.

The service label which the backup egress router advertises to the protector can be the same as the label which the backup egress router advertises to the ingress router(s), if and only if the forwarding state of the label does not direct service packets towards the protected egress router. Otherwise, the label is not usable for egress protection, because it will create a loop, which MUST be avoided. In this case, the backup egress router MUST advertise a unique service label for egress protection, and set its forwarding state to use the backup egress router’s connectivity with the service destination.

6. Egress link protection

Egress link protection is achievable through procedures similar to that of egress node protection. In normal situations, an egress router forwards service packets to a service destination based on a service label, whose forwarding state points to an egress link. In egress link protection, the egress router acts as PLR, by performing local failure detection and local repair. Specifically, the egress router pre-establishes an egress-protection bypass tunnel to a protector, and installs bypass forwarding state for the service label, pointing to the bypass tunnel. During local repair, the egress router reroutes service packets via the bypass tunnel to the protector. The protector in turn forwards the packets to the service destination (in the co-located protector mode, as shown in Figure-3), or forwards the packets to a backup egress router (in the centralized protector mode, as shown in Figure-4).

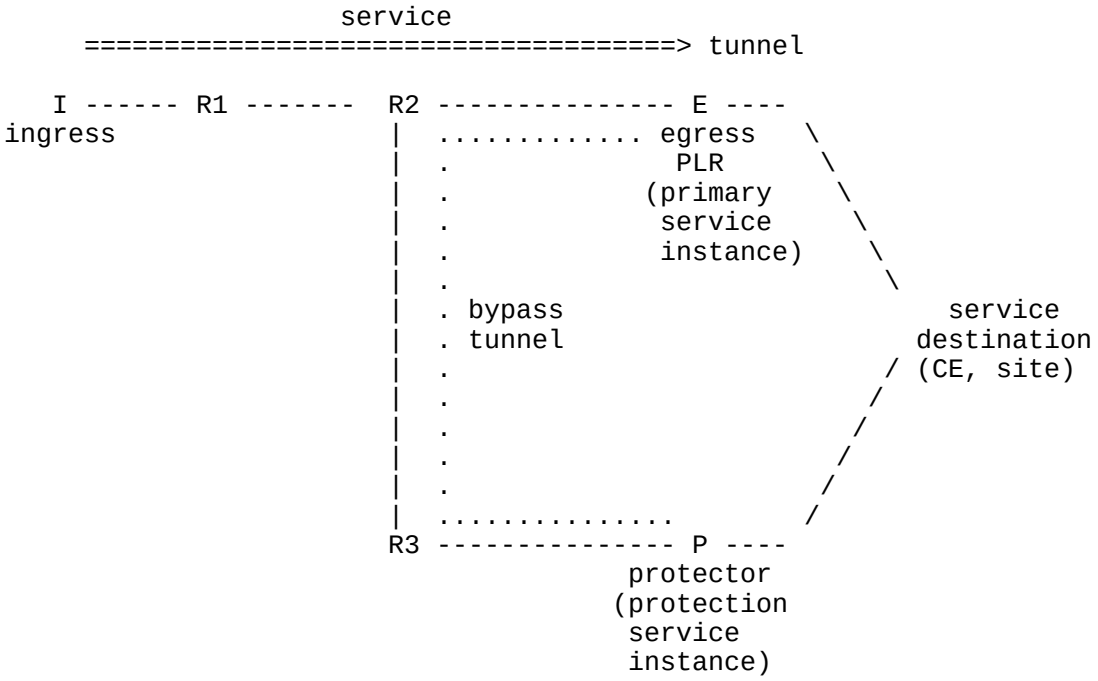

service
tunnel
I
ingress
R1
R2
E
egress
PLR
(primary
service
instance)
bypass
tunnel
service
destination
(CE, site)
R3
P
protector
(protection
service
instance)

Figure 3

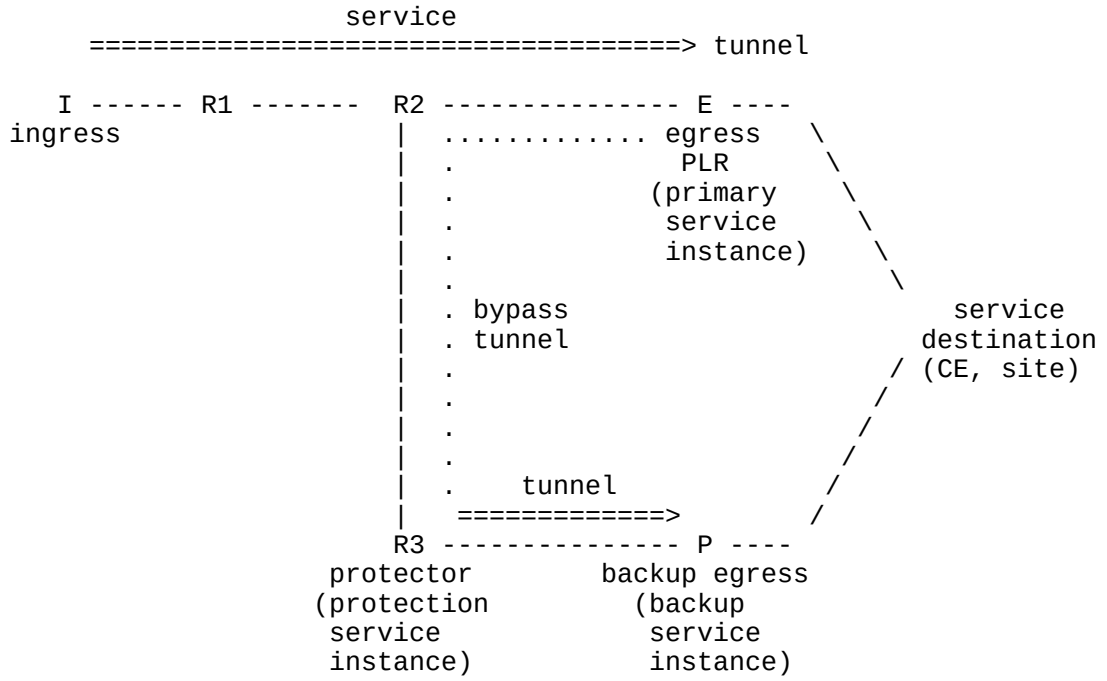


Figure 4

There are two approaches to set up the bypass forwarding state on the egress router, depending on whether the egress router knows the service label advertised by the backup egress router. The difference is that one approach requires the protector to perform context label switching, and the other one does not. Both approaches are equally supported by this framework, and may be used in parallel.

(1) The first approach applies when the egress router does not know the service label advertised by the backup egress router. In this case, the egress router sets up the bypass forwarding state as a label push with the outgoing label of the egress-protection bypass tunnel. Rerouted packets will have the egress router's service label intact. Therefore, the protector MUST perform context label switching, and the bypass tunnel MUST be destined for the context ID of the {E, P} and established as described in Section 5.9. This approach is consistent with egress node protection. Hence, a protector can serve in egress node and egress link protection in a consistent manner, and both the co-located protector mode and the centralized protector mode may be used (Figure-3 and Figure-4).

(2) The second approach applies when the egress router knows the service label advertised by the backup egress route, via a label distribution protocol session. In this case, the backup egress router serves as the protector for egress link protection, regardless of the protector of egress node protection, which should be the same router in the co-located protector mode but may be a different router in the centralized protector mode. The egress router sets up the bypass forwarding state as a label swap from the incoming service label to the service label of the protector, followed by a push with the outgoing label (or label stack) of the egress link protection bypass tunnel. The bypass tunnel is a regular tunnel destined for an IP address of the protector, instead of the context ID of the {E, P}. The protector simply forwards rerouted service packets based on its own service label, rather than performing context label switching. With this approach, only the co-located protector mode is applicable.

Note that for a bidirectional service, the physical link of an egress link may carry service traffic bi-directionally. Therefore, an egress link failure may simultaneously be an ingress link failure for the traffic in the opposite direction. However, protection for ingress link failure SHOULD be provided by a separate mechanism, and hence is out of the scope of this document.

7. Global repair

This framework provides a fast but temporary repair for egress node and egress link failures. For permanent repair, it is RECOMMENDED that the traffic SHOULD be moved to an alternative tunnel or alternative services which are fully functional. This is referred to as global repair. Possible triggers of global repair include control plane notifications of tunnel and service status, end-to-end OAM and fault detection at tunnel or service levels, and others. The alternative tunnel and services may be pre-established as standby, or dynamically established as a result of the triggers or network protocol convergence.

8. Example: Layer-3 VPN egress protection

This section shows an example of egress protection for a layer-3 VPN.

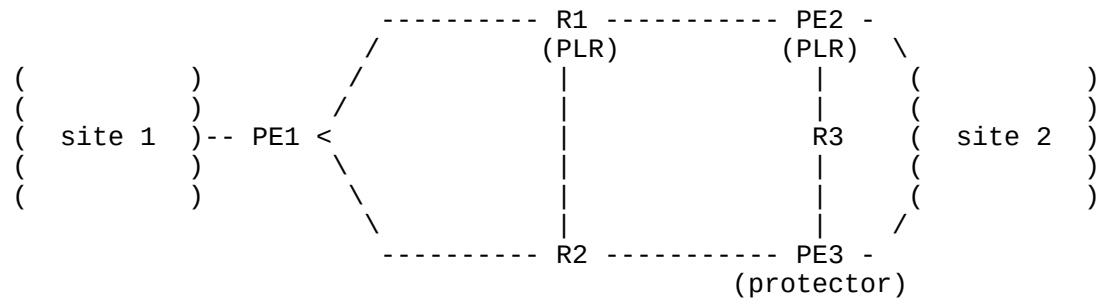


Figure 5

In this example, the site 1 (subnet 203.0.113.192/26) of a given VPN is attached to PE1, and site 2 (subnet 203.0.113.128/26) is dual-homed to PE2 and PE3. PE2 is the primary PE for site 2, and PE3 is the backup PE. Each PE hosts a VPN instance. R1 and R2 are transit routers in the MPLS network. The network uses OSPF as routing protocol, and RSVP-TE as tunnel signaling protocol. The PEs use BGP to exchange VPN prefixes and VPN labels between each other.

Using the framework in this document, the network assigns PE3 to be a protector for PE2 to protect the VPN traffic in the direction from site 1 to site 2. This is the co-located protector mode. Hence, PE2 and PE3 form a protected egress {PE2, PE3}. A context ID 198.51.100.1 is assigned to the protected egress {PE2, PE3}. The VPN instance on PE3 serves as a protection instance for the VPN instance on PE2. On PE3, a context label 100 is assigned to the context ID, and a label table pe2.mpls is created to represent PE2’s label space. PE3 installs the label 100 in its default MPLS forwarding table, with nexthop pointing to the label table pe2.mpls. PE2 and PE3 are coordinated to use the proxy mode to advertise the context ID in the routing domain and the TE domain.

PE2 uses per-VRF VPN label allocation mode. It assigns a single label 9000 to the VRF of the VPN. For a given VPN prefix 203.0.113.128/26 in site 2, PE2 advertises it along with the label 9000 and other attributes to PE1 and PE3 via BGP. In particular, the NEXT_HOP attribute is set to the context ID 198.51.100.1.

Similarly, PE3 also uses per-VRF VPN label allocation mode. It assigns a single label 10000 to the VRF of the VPN. For the VPN prefix 203.0.113.128/26 in site 2, PE3 advertises it along with the label 10000 and other attributes to PE1 and PE2 via BGP. In particular, the NEXT_HOP attribute is set to an IP address of PE3.

Upon receipt and acceptance of the BGP advertisement, PE1 uses the context ID 198.51.100.1 as destination to compute a TE path for an egress-protected tunnel. The resulted path is PE1->R1->PE2. PE1 then uses RSVP to signal the tunnel, with the context ID 198.51.100.1 as destination, and with the "node protection desired" flag set in the SESSION_ATTRIBUTE of RSVP Path message. Once the tunnel comes up, PE1 maps the VPN prefix 203.0.113.128/26 to the tunnel and installs a route for the prefix in the corresponding VRF. The route's nexthop is a push with the VPN label 9000, followed by a push with the outgoing label of the egress-protected tunnel.

Upon receipt of the above BGP advertisement from PE2, PE3 (i.e. the protector) recognizes the context ID 198.51.100.1 in the NEXT_HOP attribute, and installs a route for label 9000 in the label table pe2.mpls. PE3 sets the route's nexthop to a "protection VRF". This protection VRF contains IP routes corresponding to the IP prefixes in the dual-homed site 2, including 203.0.113.128/26. The nexthops of these routes MUST be based on PE3's connectivity with site 2, even if this connectivity is not the best path in PE3's VRF due to metrics (e.g. MED, local preference, etc.), and MUST NOT use any path traversing PE2. Note that the protection VRF is a logical concept, and it may simply be PE3's own VRF if the VRF satisfies the requirement.

8.1. Egress node protection

R1, i.e. the penultimate-hop router of the egress-protected tunnel, serves as the PLR for egress node protection. Based on the "node protection desired" flag and the destination address (i.e. context ID 198.51.100.1) of the tunnel, R1 computes a bypass path to 198.51.100.1 while avoiding PE2. The resulted bypass path is R1->R2->PE3. R1 then signals the path (i.e. egress-protection bypass tunnel), with 198.51.100.1 as destination.

Upon receipt of an RSVP Path message of the egress-protection bypass tunnel, PE3 recognizes the context ID 198.51.100.1 as the destination, and hence responds with the context label 100 in an RSVP Resv message.

After the egress-protection bypass tunnel comes up, R1 installs a bypass nexthop for the egress-protected tunnel. The bypass nexthop is a swap from the incoming label of the egress-protected tunnel to the outgoing label of the egress-protection bypass tunnel.

When R1 detects a failure of PE2, it will invoke the above bypass nexthop to reroute VPN service packets. The packets will have the label of the bypass tunnel as outer label, and the VPN label 9000 as inner label. When the packets arrive at PE3, they will have the

context label 100 as outer label, and the VPN label 9000 as inner label. The context label will first be popped, and then the VPN label will be looked up in the label table pe2.mpls. The lookup will cause the VPN label to be popped, and the IP packets will finally be forwarded to site 2 based on the protection VRF.

8.2. Egress link protection

PE2 serves as the PLR for egress link protection. It has already learned the VPN label 10000 from PE3, and hence it uses the approach (2) described in Section 6 to set up bypass forwarding state. It signals an egress-protection bypass tunnel to PE3, by using the path PE2->R3->PE3, and PE3's IP address as destination. After the bypass tunnel comes up, PE2 installs a bypass nexthop for the VPN label 9000. The bypass nexthop is a label swap from the incoming label 9000 to the VPN label 10000 of PE3, followed by a label push with the outgoing label of the bypass tunnel.

When PE3 detects a failure of the egress link, it will invoke the above bypass nexthop to reroute VPN service packets. The packets will have the label of the bypass tunnel as outer label, and the VPN label 10000 as inner label. When the packets arrive at PE3, the VPN label 10000 will be popped, and the IP packets will be forwarded based on the VRF indicated by on the VPN label 10000.

8.3. Global repair

Eventually, global repair will take effect, as control plane protocols converge on the new topology. PE1 will choose PE3 as new entrance to site 2. Before that happens, the VPN traffic has been protected by the above local repair.

9. IANA Considerations

This document has no request for new IANA allocation.

10. Security Considerations

The framework in this document relies on fast reroute around a network failure. Specifically, service traffic is temporarily rerouted from a PLR to a protector. In the centralized protector mode, the traffic is further rerouted from the protector to a backup egress router. Such kind of fast reroute is planned and anticipated, and hence it should not be viewed as a new security threat.

The framework requires a service label distribution protocol to run between an egress router and a protector. The available security

measures of the protocol MAY be used to achieve a secured session between the two routers.

11. Acknowledgements

This document leverages work done by Yakov Rekhter, Kevin Wang and Zhaohui Zhang on MPLS egress protection. Thanks to Alexander Vainshtein, Rolf Winter, and Lizhong Jin for their valuable comments that helped shape this document and improve its clarity.

12. References

12.1. Normative References

[SR-ARCH] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing (work in progress), 2017.

[SR-OSPF] Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions (work in progress), 2017.

[SR-ISIS] Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions (work in progress), 2017.

12.2. Informative References

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

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <https://www.rfc-editor.org/info/rfc7490>.

[RFC7812] Atlas, A., Bowers, C., and G. Enyedi, "An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7812, DOI 10.17487/RFC7812, June 2016, <https://www.rfc-editor.org/info/rfc7812>.

[RFC8104] Shen, Y., Aggarwal, R., Henderickx, W., and Y. Jiang, "Pseudowire (PW) Endpoint Fast Failure Protection", RFC 8104, DOI 10.17487/RFC8104, March 2017, <https://www.rfc-editor.org/info/rfc8104>.

[BGP-PIC] Bashandy, P., Filsfils, C., and P. Mohapatra, "BGP Prefix Independent Convergence", draft-ietf-rtgwg-bgp-pic-05.txt (work in progress), 2017.

[RSVP-EP] Chen, H., Liu, A., Saad, T., Xu, F., Huang, L., and N. So, "Extensions to RSVP-TE for LSP Egress Local Protection", draft-ietf-teas-rsvp-egress-protection (work in progress), 2017.

Authors' Addresses

Yimin Shen
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Phone: +1 9785890722
Email: yshen@juniper.net

Minto Jeyananth
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Phone: +1 4089367563
Email: minto@juniper.net

Bruno Decraene
Orange

Email: bruno.decraene@orange.com

Hannes Gredler
RtBrick Inc

Email: hannes@rtbrick.com

Carsten Michel
Deutsche Telekom

Email: c.michel@telekom.de

Huaimo Chen
Huawei Technologies Co., Ltd.

Email: huaimo.chen@huawei.com

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129
China

Email: jiangyuanlong@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 24, 2021

X. Xu
Alibaba, Inc.
H. Assarpour
Broadcom
S. Ma
Mellanox
F. Clad
Cisco Systems, Inc.
December 21, 2020

MPLS Payload Protocol Identifier
draft-xu-mpls-payload-protocol-identifier-08

Abstract

The MPLS label stack has no explicit protocol identifier field to indicate the protocol type of the MPLS payload. This document proposes a mechanism for containing a protocol identifier field within the MPLS packet, which is useful for any new encapsulation header (e.g., INT metadata) which may need to be encapsulated with an MPLS header.

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Terminology 3
3. Protocol Type Field 3
4. Data Plane Processing of PIL 4
4.1. Egress LSRs 4
4.2. Ingress LSRs 4
4.3. Transit LSRs 5
4.4. Penultimate Hop LSRs 5
5. Signaling for PIL Processing Capability 5
6. Alternative Approaches 5
7. Acknowledgements 6
8. IANA Considerations 6
9. Security Considerations 6
10. References 6
10.1. Normative References 6
10.2. Informative References 6
Authors' Addresses 7

1. Introduction

The MPLS label stack has no explicit protocol identifier field to indicate the protocol type of the MPLS payload. This document proposes a mechanism for containing a protocol identifier field within the MPLS packet, which is useful for any new encapsulation header (e.g., INT metadata) which may need to be encapsulated with an MPLS header. With this explicit protocol identifier field, there is no need any more for each new encapsulation header to deal with the notorious first nibble issue associated with MPLS individually. More specifically, there is no need to intentionally avoid the first nibble of each new encapsulation header from being 0100 (IPv4) or 0110 (IPv6) and even worsely misuse the first nibble of each new encapsulation header as an MPLS payload type field (e.g., MPLS-BIER [RFC8296]). The tacit permission of misusing the first nibble of each new encapsulation header as an MPLS payload type field would exhause the valuable nibble space quickly. Furthermore, there is no need to insert one additional label indicating the MPLS payload type when transporting any new encapsulation header over MPLS LSPs (e.g., transporting Network Service Header (NSH) [RFC8300] over MPLS LSPs)

therefore the signalling for that additional label is not needed anymore.

To some extent, this situation is much similar to that of the MPLS reserved label space (a.k.a., the special purpose label space) [RFC7274] . Due to the concern over the scarcity of the special-purpose label space , the extended special purpose label concept is introduced accordingly. Similarily, the IETF MPLS community should take precautions on the the scarcity of the first nibble of the MPLS payload before it is too late.

1.1. Requirements Language

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

2. Terminology

This memo makes use of the terms defined in [RFC3032].

3. Protocol Type Field

The encapsulation format for Protocol Type field is depicted as below:

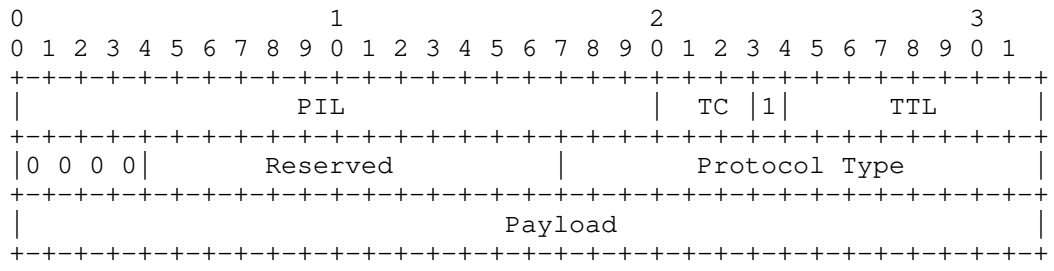


Figure 1

Protocol Identifier Label (PIL): This field contains a special purpose label with value of <TBD> or an extended special purpose label [RFC7274] with value of <TBD> which indicates that a Protocol Type field appears immediately after the bottom of the label stack.

Traffic Class (TC): The usage of this field is in accordance with the current MPLS specification [RFC3032].

S: The Bottom of Stack (BoS) field is set since the PIL MUST always appear at the bottom of the label stack.

TTL: The usage of this field is in accordance with the current MPLS specification [RFC3032].

Reserved MUST be set to 0 and ignored on reception.

Protocol Type: This field indicates the protocol type of the MPLS payload as per [ETYPES].

Payload: This field contains the MPLS payload which can be an IP packet, an Ethernet frame, or any other type of payload, e.g., Network Service Header (NSH) [RFC8300].

4. Data Plane Processing of PIL

4.1. Egress LSRs

Suppose egress LSR Y is capable of processing the Protocol Type field contained in MPLS packets. LSR Y indicates this to all ingress LSRs via signaling (see Section 5). LSR Y MUST be prepared to deal with both packets with an imposed Protocol Type field and those without; the PIL will distinguish these cases. If a particular ingress LSR chooses not to impose a Protocol Type field, LSR Y's processing of the received label stack (which might be empty) is as if LSR Y chose not to accept Protocol Type field. If an ingress LSR X chooses to impose the Protocol Type field, then LSR Y will receive an MPLS packet constructed as follows: <Top Label (TL), Application Label (AL), PIL> <Protocol Type field> <remaining MPLS payload>. Note that here the TL could be replaced with an IP-based tunnel [RFC4023] and the AL is optional. LSR Y recognizes TL as the label it distributed to its upstream LSR and pops the TL (note that the TL may be an implicit null label, in which case it doesn't appear in the label stack and LSR Y MUST process the packet starting with the AL label (if present) and/or the PIL.) LSR Y recognizes the PIL with S bit set. LSR Y then processes the Protocol Type field, which will determine how LSR Y processes the MPLS payload.

4.2. Ingress LSRs

If an egress LSR Y indicates via signaling that it can process the Protocol Type field, an ingress LSR X can choose whether or not to insert it into the MPLS packet destined for LSR Y. The ingress LSR X MUST NOT insert the Protocol Type field into that MPLS packet unless the egress LSR X has explicitly announced that it could process it. The steps that ingress LSR X performs to insert the Protocol Type field are as follows:

1. On an incoming packet, identify the application to which the packet belongs and determine whether the Protocol Type field needs to be added to the incoming packet.

2. For packets requiring the insertion of the Protocol Type field, prepend the Protocol Type field to the existing MPLS payload; then, push the PIL on to the label stack with the S bit set.

3. Push the application label (AL) label (if required) on to the label stack.

4. Push the EL and the ELI labels [RFC6790] on to the label stack (if required).

5. Determine the top label (TL) and push it on to the label stack.

6. Determine the output interface and send the packet out.

4.3. Transit LSRs

Transit LSRs MAY operate with no change in forwarding behavior. If a transit LSR recognizes the PIL and the subsequent Protocol Type field, it MAY be allowed to do some additional value-added processing, such as MPLS payload inspection, on the received MPLS packet containing the PIL and the Protocol Type field.

4.4. Penultimate Hop LSRs

No change is needed at penultimate hop LSRs.

5. Signaling for PIL Processing Capability

TBD.

6. Alternative Approaches

As illustrated in Section 3 and Section 4, the existence of the Protocol Type field immediately after the MPLS label stack is indicated by inserting the PIL into an MPLS packet. Alternatively, by setting the first nibble of the 4-octet entry containing the Protocol Type field to a dedicated value (e.g., 1111), the existence of the Protocol Type field could be indicated as well (see Figure 2). In this way, there is no need to insert additional label(s) (i.e., the PIL) into an MPLS packet. As for which approach should be selected in the end, it depends on a wide-scope discussion within the IETF.

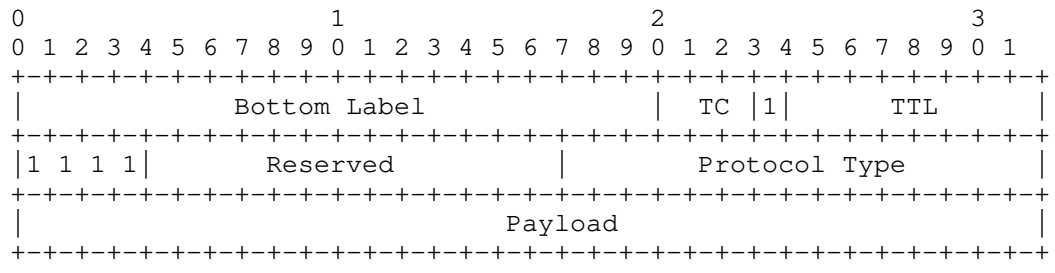


Figure 2

7. Acknowledgements

TBD.

8. IANA Considerations

A special purpose label with value of <TBD> or an extended special purpose label with value of <TBD> for the PIL needs to be assigned by the IANA

9. Security Considerations

TBD.

10. References

10.1. Normative References

[ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", 2012.

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

10.2. Informative References

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

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <https://www.rfc-editor.org/info/rfc4023>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <https://www.rfc-editor.org/info/rfc6790>.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <https://www.rfc-editor.org/info/rfc7274>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/info/rfc8296>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Authors' Addresses

Xiaohu Xu
Alibaba, Inc.

Email: 13910161692@qq.com

Hamid Assarpour
Broadcom

Email: hamid.assarpour@broadcom.com

Shaowen Ma
Mellanox

Email: mashaowen@gmail.com

Francois Clad
Cisco Systems, Inc.

Email: fclad@cisco.com