PALS Working Group
Internet Draft
Intended Status: Proposed Standard
Expires: April 26, 2015

Patrice Brissette
Kamran Raza
Sami Boutros
Cisco Systems, Inc.

Nick Del Regno
Matthew Turlington
Verizon

June 29, 2015

Handling Incoming Label Request for PW FEC Types
draft-brissette-pals-pw-fec-label-request-01

Abstract

This document clarifies the behavior of an LSR PE upon receiving an LDP Label Request message for Pseudowire (PW) FEC types. Furthermore, this document specifies the procedures to be followed by the LSR PE in order to answer such requests for a given PW FEC type.

Status of this Memo

Copyright and License Notice

Convention

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].

Table of Contents

1 Introduction

Label Distribution Protocol (LDP) base specification [RFC5036] defines different LDP message types and their procedures for advertising label bindings. These procedures are generic and inherited by any FEC type that is advertised using these message types. For a given FEC type, any difference in behavior, compared to what is already specified in RFC 5036, needs to be spelled out clearly in the corresponding specification in which the FEC type is being introduced or extended.

[RFC4447] specifies mechanisms to setup pseudowires (PWs) using LDP. [RFC4447] does not specify any behavior change with regards to label binding distribution for PW FEC types in response to a corresponding Label Request message from a peer LSR PE. This implies that [RFC4447] inherits the base procedures defined in [RFC5036] for Label Request and associated response for a PW FEC type. The lack of specification in the area of Label Request in [RFC4447] has led to some interoperability issues between vendors due to different interpretation. For example, there are some implementations which do not honor and do not respond to an incoming Label Request for a PW FEC type, resulting in functionality impact. Some of these problems are very critical for the deployment of PW technologies. The following is a non-exhaustive list of some of the problems and potential breakages that may result due to the lack of support of incoming Label Request for a PW FEC:

- An LSR PE can not restart forwarding of packet with sequence number 1 as specified in section 4.1 of [RFC4385] with regards to Control Word Sequencing.

- An LSR PE may not be able to perform a PW consistency check as defined in section 4.1 of [RFC6667], resulting in LSR PEs becoming out-of-sync.

- Some implementations of LSR PE do not checkpoint PW label bindings learnt from peer(s) in their persistent memory and hence are not able to recover any peer state after their own restarts or switchovers. Such implementations typically require re-learning of peer’s label bindings after their own failure and rely on Label Request mechanisms.

- The combination of Downstream Unsolicited mode and Conservative Label retention (used due to memory limitations) can lead to a situation where an LSR PE releases the label learnt from a peer for a PW that it may need later. Label Request is used to solve this issue. For example, consider an LSR PE operating in Label Conservative mode receiving a label binding for a non-locally configured/known PW. This LSR PE ignores such a label binding and later tries to re-learn it via Label Request procedure once PW is locally configured. The authors will like to remind the readers about the following fact: [RFC4447] does not mandate to use Label Liberal mode. Therefore it is possible that some implementation used Label Conservative mode.

This document clarifies the use of Label Request message and its procedures for PW FEC types and re-enforces the acceptable behavior to be implemented by an LSR PE.

2. Requirements

This document recommends the following action to be implemented by an LSR PE that supports a PW FEC Type (P2P or P2MP type):

- An LSR PE MUST respond to an incoming Label Request message for a PW FEC by sending its local binding for the PW via a Label Mapping message. If no such binding is available, the LSR PE SHOULD respond by sending a new status code "No PW" in a Notification message.

- An LSR PE MUST respond to an incoming Label Request message for a Wildcard FEC [RFC5036] by sending its local bindings for all its PWs via Label Mapping messages. This is in addition to label bindings corresponding to any other LDP FEC types configured and available at the LSR.

- An LSR PE MUST respond to an incoming Label Request message for a Typed Wildcard PW FEC [RFC6667] by sending its local bindings for all its PWs for the given FEC type via Label Mapping messages. For a given PW FEC type, this advertisement is to be scoped either for a specific PW type or for all PW types according to the received PW Typed Wildcard FEC.

3. Procedures

This document re-enforces the Label Request generic procedures, as defined by RFC 5036, for PW FEC types, and hence strongly recommends that an LSR PE receiving the PW Label Request message should respond either by sending its label binding in Label Mapping message(s) or with a Notification message indicating why it cannot satisfy the request.

An LSR PE should respond to a Label Request when corresponding PW FEC is resolved locally. The following sub sections define the meaning of a "resolution" for a given PW FEC type.

3.1 PWid FEC (FEC128)

A PWid FEC is resolved when a local label binding has been allocated after local configuration application.

[RFC6073] does not preclude setting up MS-PWs using FEC-128, therefore this procedure is also applicable to PEs acting as S-PEs.

3.2 Generalized PWid FEC (FEC129):

A Generalized PWid FEC is resolved at an ST-PE when SAII is locally configured, TAII is learnt statically or dynamically via discovery mechanisms, and a local label binding has been allocated.

This FEC is resolved at an TT-PE when SAII is locally configured, TAII is learnt statically or dynamically via discovery mechanisms, remote label binding is received, and a local label binding has been allocated.

Whereas, this FEC is resolved at an S-PE when remote label binding is received for PW segment, TAII is learnt statically or dynamically via discovery mechanisms, and a local label binding has been allocated.

3.3 Common to PWid and Generalized PWid FEC

A FEC is resolved at an S-PE when remote label binding is received for PW segment.

In the case of Generalized PWid FEC, TAII is learnt statically or dynamically via discovery mechanisms, and a local label binding has been allocated. Whereas PWid FEC is resolved when a local binding has been allocated.

3.4 P2MP PW Upstream FEC (FEC130):

Editor Note: Deferred for further study.

3.5 P2MP PW Downstream FEC (FEC132):

Editor Note: Deferred for further study.

3.5 PW Typed Wildcard FEC

The rules defined for individual PW FEC types apply equally when they are used under a PW Typed Wildcard FEC [RFC6667].

4 Acknowledgements

The authors would like to thank for Alexander Vainshtein its reviews and comments of this document.

5 Security Considerations

This document does not introduce any additional security constraints.

6 IANA Considerations

This document requires the assignment of a new LDP Status Code to be used in a Notification message to notify a peer LSR if lookup fails at receiving LSR for a PW FEC received in a Label Request message.

The value requested from the IANA managed LDP registry "LDP Status Code Name Space" is:

Range/Value	E	Description
0x00000032	0	No PW

7 References

7.1 Normative References

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

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

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

[RFC6667] Raza, K., Boutros, S., and Pignataro, C., "LDP Typed Wildcard FEC for PWid and Generalized PWid FEC", RFC 6667, July 2012.

7.2 Informative References

Authors' Addresses

Patrice Brissette
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8, Canada.
EMail: pbrisset@cisco.com

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8, Canada.
EMail: skraza@cisco.com

Sami Boutros
Cisco Systems, Inc.
3750 Cisco Way,
San Jose, CA 95134, USA.
E-mail: sboutros@cisco.com

Nick Del Regno

Verizon
400 International Pkwy
Richardson, TX 75081, USA.
E-mail: nick.delregno@verizon.com

Matthew Turlington
Verizon
400 International Pkwy
Richardson, TX 75081, USA.
E-mail: matt.turlington@verizon.com

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

W. Cheng
L. Wang
H. Li
China Mobile
K. Liu
Huawei Technologies
S. Davari
Broadcom Corporation
J. Dong
Huawei Technologies
A. D'Alessandro
Telecom Italia

October 27, 2014

Dual-Homing Coordination for MPLS Transport Profile (MPLS-TP) Pseudowires

draft-cheng-pwe3-mpls-tp-dual-homing-coordination-00

Abstract

In some scenarios, the MPLS Trasport Profile (MPLS-TP) Pseudowires (PWs) are provisioned through either static configuration or management plane, where a dynamic control plane is not available. A fast protection mechanism for MPLS-TP PWs is needed to protect against the failure of Attachment Circuit (AC), the failure of Provider Edge (PE) and also the failure in the Packet Switched Network (PSN). The framework and scenarios for dual-homing pseudowire (PW) local protection are described in [draft-cheng-pwe3-mpls-tp-dual-homing-protection]. This document proposes a dual-homing coordination mechanism for MPLS-TP PWs, which is used for state exchange and coordination between the dual-homing PEs for dual-homing PW local protection.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

[RFC6372] and [RFC6378] describe the framework and mechanism of MPLS-TP Linear protection, which can provide protection for the MPLS LSP or PW between the edge nodes. Such mechanism does not protect the failure of the Attachement Circuit (AC) or the endpoint nodes.

In some scenarios such as mobile backhauling, the MPLS PWs are provisioned with dual-homing topology, in which at least the CE node in one side is dual-homed to two PEs. If a failure occurs in the primary AC, operators usually prefer to perform switchover only in the dual-homing PE side and keep the working pseudowire unchanged if possible. This is to avoid massive PW switchover in the mobile backhaul network due to the AC failure in the core site, and also could achieve efficient and balanced link bandwidth utilization. Similarly, it is preferable to keep using the working AC when one working PW fails in PSN network. A fast dual-homing PW protection mechanism is needed to protect the failure in AC, the the PE node and the PSN network to meet the above requirements.

[I-D.cheng-pwe3-mpls-tp-dual-homing-protection] describes a framework and several scenarios for dual-homing pseudowire (PW) local protection. This document proposes a dual-homing coordination mechanism for static MPLS-TP PWs, which is used for information exchange and coordination between the dual-homing PEs for the dual-homing PW local protection. The proposed mechanism has been deployed in several mobile backhaul networks which use static MPLS-TP PWs for the backhauling of mobile traffic from the RF sites to the core site.

2. Overview of the Proposed Solution

The linear protection mechanisms for MPLS-TP network are defined in [RFC6378], [RFC7271] and [RFC7324]. When such mechanisms are applied to PW linear protection, both the working PW and the protection PW terminate on the same PE nodes. In order to provide dual-homing protection for MPLS-TP PWs, some additional mechanisms are needed.

In MPLS-TP PW dual-homing protection, the linear protection mechanisms on the single-homing PE (e.g. PE3 in figure 3) are not changed, while on the dual-homing side, the working PW and protection PW are terminated on two dual-homing PEs (e.g. PE1 and PE2 in figure 1) respectively to protect the failure occurs in the dual-homing PEs and the connected ACs. As specified in [I-D.cheng-pwe3-mpls-tp-dual-homing-protection], a dedicated Dual-Node Interconnection (DNI) PW is provisioned between the two dual-homing PE nodes, which is used to bridge the traffic between the dual-homing PEs when failure happens in the working PW or the primary AC. In order to make the linear protection mechanism work in the dual-homing PEs scenario, some coordination between the dual-homing PE nodes is needed, so that the dual-homing PEs can set the connection between AC, the service PW and the DNI-PW properly in a coordinated fasion.

```
               +----------------+
               /                |                +--------+
          AC1 /|   PE1          |  Working PW    |        |
             / |                X----------------X        |
            /  |                |  Service PW1   |        |
       +---/+  +--------X-------+                |        |   +----+
       |    |           | DNI PW                 |  PE3   |   |    |
       | CE1|           |                        |        |---| CE2|
       +---\+  +--------X-------+                |        |   +----+
            \  |                | Protection PW  |        |
             \ |                X----------------X        |
          AC2 \|                | Service PW2    |        |
               \   PE2          |                +--------+
               +----------------+
```

Figure 1. Dual-homing Proctection with DNI-PW

3. Protocol Extensions for MPLS-TP PW Dual-Homing Protection

In dual-homing MPLS-TP PW local protection, the forwarding state of the dual-homing PEs are determined by the forwarding state machine as defined in [I-D.cheng-pwe3-mpls-tp-dual-homing-protection]. In order to achieve the MPLS-TP PW dual-homing protection, coordination between the dual-homing PE nodes is needed to exchange the PW status and protection coordination requests.

3.1. Information Exchange Between Dual-Homing PEs

The coordination information will be sent over the G-ACh as described in [RFC5586]. A new G-ACh channel type is defined for the coordination between the dual-homing PEs of MPLS-TP PWs. This channel type can be used for the exchange of different kinds of information between the dual-homing PEs. This document uses this channel type for the PW status exchange and switchover coordination between the dual-homing PEs. Other potential usage of this channel type are for further study and are out of the scope of this document.

The MPLS-TP Dual-Homing Coordination (DHC) message is sent on the DNI PW between the dual-homing PEs. The format of MPLS-TP DHC message is shown below:

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0 0 0 1|Version|     Flags     |         DHC Code Point        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Dual-Homing Group ID                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         TLV  Length           |           Reserved            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ~                              TLVs                             ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. MPLS-TP Dual-Homing Coordination Message

The Dual-Homing Group ID is a 4-octet unsigned integer to identify the dual-homing PEs in the same dual-homing group.

In this document, 2 TLVs are defined in MPLS-TP Dual-Homing Coordination message for dual-homing MPLS-TP PW protection:

Type	Description	Length
1	PW Status	20 Bytes
2	Dual-Node Switching	16 Bytes

The PW Status TLV is used by a dual-homing PE to report its service PW status to the other dual-homing PE in the same dual-homing group.

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     Type=1 (PW Status)        |          Length               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                     Destination Node_ID                       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                       Source Node_ID                          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         DNI PW-ID                             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         Reserved                            |P|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                       Service PW State                    |D|F|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3. PW Status TLV

- The Destination Node_ID is the 32-bit Node_ID of the receiver PE.

- The Source Node_ID is the 32-bit Node_ID of the sending PE.

- The DNI PW-ID field contains the 32-bit PW-ID of the DNI PW.

- The P (Protection) bit indicates whether the message is sent by the working PE (P=0) or by the protection PE (P=1).

- The Service PW State field indicates the state of the Service PW between the sending PE and the remote PE. Currently two bits are defined in the Service PW Request field:

o F bit: Indicates Signal Fail (SF) is generated on the service PW. It can be either a local request or a remote request received from the remote PE.

o D bit: Indicates Signal Degrade (SD) generated on the service PW. It can be either a local request or a remote request received from the remote PE.

o Other bits are reserved and MUST be set to 0 on transmission and SHOULD be ignored upon receipt.

The Dual-Node Switching TLV is used by the protection dual-homing PE to send protection state coordination to the working dual-homing PE.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type=2 (Dual-Node Switching) |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination Node_ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Node_ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         DNI PW-ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Reserved                         |S|P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4. Dual-node Switching TLV

- The Destination Node_ID is the 32-bit Node_ID of the receiving PE.

- The Source Node_ID is the 32-bit Node_ID of the sending PE.

- The DNI PW-ID field contains PW-ID of the DNI PW.

- The P (Protection) bit indicates whether the message is sent by the working PE (P=0) or by the protection PE (P=1). With the mechanism described in this document, only the protection PE could send DHC message with the Dual-node Switching TLV.

- The S (PW Switching) bit indicates which service PW is used for transporting user traffic. It is set to 0 when traffic is transported on the working PW, and is set to 1 if traffic will be transported on the protection PW. The value of the S bit is determined by the protection coordination mechanism between the dual-homing protection PE and the remote PE.

The MPLS-TP DHC message is exchanged periodically between the dual-homing PEs. Whenever there is a change in the status of service PW on one dual-homing PE, it MUST be sent to the other dual-homing PE immediately using the PW status TLV in the DHC message. The Dual-Node Switching TLV is carried in the DHC message when a switchover request is issued by the protection PE according to the dual-homing forwarding state machine.

3.2. Protection Procedures

The dual-homing MPLS-TP PW protection mechanism can be deployed with the existing AC redundancy mechanisms, e.g. Multi-Chassis Link Aggregation Group (MC-LAG). On the PSN network side, PSN tunnel protection mechanism is not required, as the dual-homing PW protection can also protect the failure occurs in the PSN network.

On the single-homing PE side, it just treats the working PW and protection PW as if they terminate on the same remote PE node, thus normal MPLS-TP protection coordination mechanisms still apply to the single-homing PE.

The forwarding behavior of the dual-homing PEs is determined by the components shown in the figure below:

```
             +----------------------------------+           +-----+
             |        PE1 (Working PE)          |           |     |
             +----------------------------------+           |     |
             |                 |                |    PW1    |     |
             +    Forwarder    +      Service   X<--------->X     |
            /|                 |        PW      |  Working  |     |
           / +--------+--------+                |    PW     |     |
     AC1  /  |     DNI PW      |                |           |     |
         /   +--------X--------+----------------+           |     |
  +-----+/            ^                                     |     |
  | CE1 |             |  DNI PW                             | PE3 | +---+
  +-----+             |                                     |     --|CE3|
        \             V                                     |     | +---+
     AC2 \   +--------X--------+----------------+           |     |
          \  |     DNI PW      |                |           |     |
           \ +--------+--------+                |           |     |
            \|                 |      Service   |    PW2    |     |
             +    Forwarder    +        PW      X<--------->X     |
             |                 |                | Protection|     |
             +----------------------------------+    PW     |     |
             |        PE2 (Protection PE)       |           |     |
             +----------------------------------+           +-----+
```

Figure 5. Components of PW dual-homing protection

In figure 5, for a dual-homing PE, service PW is the PW used to carriy service between the dual-homing PE and the remote PE. The status of service PW is determined by the OAM mechanism between the dual-homing PE and the remote PE.

DNI PW is the PW established between the two dual-homing PE nodes. It is used to bridge traffic when failure occurs in the PSN network or in the ACs. The status of DNI PW is determined by OAM mechanism running between the dual-homing PEs. Since DNI PW is used to carry both the coordination messages and service traffic, it is RECOMMENDED to provision multiple links between the dual-homing PEs.

AC is the link which connects the dual-homing PEs to the dual-homed CE. The status of AC is determined by MC-LAG or other AC redundancy mechanisms.

In order to perform dual-homing PW local protection, the service PW status and protection coordination requests need to be exchanged between the dual-homing PEs using the DHC message defined above.

Whenever there is a change in the status of service PW on the dual-homing PE, it MUST be sent to the other dual-homing PE in the same dual-homing group immediately using the PW status TLV in the DHC message. After the exchange of PW status, both the dual-homing PEs could obtain the status of the working and protection service PWs. The status of DNI PW is determined by the OAM mechanisms between the dual-homing PEs, and the status of AC is determined by the AC redundancy mechansim. The protection PE SHOULD make the switchover decision acording to the status of the connected AC, service PW and DNI PW, and SHOULD send the switchover request to the working PE using the Dual-node switching TLV in the DHC message. The forwarding behavoir of the dual-homing PE nodes is determined by the forwarding state machine as shown in the following table:

Service PW	AC	DNI PW	Forwarding Behavior
Active	Active	Up	Service PW <-> AC
Active	Standby	Up	Service PW <-> DNI PW
Standby	Active	Up	DNI PW <-> AC
Standby	Standby	Up	Drop all packets

Table 1. Dual-homing PE Forwarding State Machine

Using the topology in figure 5 as an example, in normal state, the working PW (PW1) is in active state, the protection PW (PW2) is in standby state, the DNI PW is up, and AC1 is in active state according to AC side redundancy mechanism. According to Table 1, traffic will be forwarded through the working PW (PW1) and the primary AC (AC1). No traffic will go through the protection PE (PE2) or the DNI PW, as both the protection PW (PW2) and the AC connecting to PE2 are in standby state.

If some failure occurs in AC1, the state of AC2 changes to active according to the AC redundancy mechanism, while there is no change in the status of the working and protection PW. According to the forwarding state machine in Table 1, PE1 starts to forward traffic between the working PW and the DNI PW, while PE2 starts to forward traffic between AC2 and the DNI PW. It should be noted that in this case only AC switchover takes place, in PSN network the traffic is still fowarded using the working PW, PW switchover is not needed.

If some failure occurs in the PSN network which causes PW1 down, the working PE (PE1) or the remote PE (PE3) can detect the failure using MPLS-TP OAM mechanism. If PE1 detects the failure, it MUST inform PE2 the status of the working PW using the PW Status TLV in MPLS-TP DHC message. According to the forwarding state machine in Table 1, PE2 SHOULD set the connection between PW2 and the DNI PW, and PE1 SHOULD set the connection between the DNI PW and AC1. For switchover coordination, PE2 MUST send a DHC message to PE1 with the S bit in the Dual-node switching TLV set, and send an appropriate protection coordination message on the protection PW (PW2) to PE3 for the remote side switchover from PW1 to PW2. Upon receipt of Dual-node switching TLV in the DHC message, PE1 MUST switch the traffic onto the connection between DNI PW and AC1. If PE3 detects the failure in PW1, it would send a protection coordination message on the protection PW (PW2) to inform PE2 to switchover to the protection PW. And PE2 MUST send a DHC message to PE1 with the S bit in the Dual-node switching TLV set to coordinate the switchover on PE1 and PE2.

If some failure causes the working PE (PE1) down, both the remote PE(PE3) and the protection PE(PE2) would detect the failure using MPLS-TP OAM mechanisms. The status of AC1 changes to standby, and the state of AC2 changes to active according to AC redundancy mechansim. PE3 would send a protection coordination message on the protection path to inform its peer node (PE2) to switchover to the protection PW. According to the forwarding state machine in Table 1, PE2 starts to forward traffic between the protection PW and AC2.

4. IANA Considerations

IANA needs to assign one new channel type for "MPLS-TP Dual-Homing Coordination messgae" from the "Pseudowire Associated Channel Types" registry.

This document creates a new registry called "MPLS-TP DHC TLVs" registry. 2 new TLVs are defined in this document:

Type	Description	Length
1	PW Status	20 Bytes
2	Dual-Node Switching	16 Bytes

5. Security Considerations

Procedures and protocol extensions defined in this document do not affect the security model of MPLS-TP linear protection as defined in [RFC6378]. Please refer to [RFC5920] for MPLS security issues and generic methods for securing traffic privacy and integrity.

6. References

6.1. Normative References

[I-D.cheng-pwe3-mpls-tp-dual-homing-protection]
Cheng, W., Wang, L., Li, H., Liu, K., Davari, S., and J. Dong, "Dual-Homing Protection for MPLS Transport Profile (MPLS-TP) Pseudowires", draft-cheng-pwe3-mpls-tp-dual-homing-protection-00 (work in progress), July 2014.

[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.

[RFC6372] Sprecher, N. and A. Farrel, "MPLS Transport Profile (MPLS-TP) Survivability Framework", RFC 6372, September 2011.

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

[RFC7271] Ryoo, J., Gray, E., van Helvoort, H., D'Alessandro, A., Cheung, T., and E. Osborne, "MPLS Transport Profile (MPLS-TP) Linear Protection to Match the Operational Expectations of Synchronous Digital Hierarchy, Optical Transport Network, and Ethernet Transport Network Operators", RFC 7271, June 2014.

[RFC7324] Osborne, E., "Updates to MPLS Transport Profile Linear Protection", RFC 7324, July 2014.

6.2. Informative References

[I-D.ietf-pwe3-endpoint-fast-protection]
Shen, Y., Aggarwal, R., Henderickx, W., and Y. Jiang, "PW Endpoint Fast Failure Protection", draft-ietf-pwe3-endpoint-fast-protection-01 (work in progress), July 2014.

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

[RFC6718] Muley, P., Aissaoui, M., and M. Bocci, "Pseudowire Redundancy", RFC 6718, August 2012.

[RFC6870] Muley, P. and M. Aissaoui, "Pseudowire Preferential Forwarding Status Bit", RFC 6870, February 2013.

Authors' Addresses

Weiqiang Cheng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: chengweiqiang@chinamobile.com

Lei Wang
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: Wangleiyj@chinamobile.com

Han Li
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: Lihan@chinamobile.com

Kai Liu
Huawei Technologies
Huawei Base, Bantian, Longgang District
Shenzhen 518129
China

Email: alex.liukai@huawei.com

Shahram Davari
Broadcom Corporation
3151 Zanker Road
San Jose 95134-1933
United States

Email: davari@broadcom.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Alessandro D'Alessandro
Telecom Italia
via Reiss Romoli, 274
Torino 10148
Italy

Email: alessandro.dalessandro@telecomitalia.it

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

W. Cheng
L. Wang
H. Li
China Mobile
K. Liu
Huawei Technologies
S. Davari
Broadcom Corporation
J. Dong
Huawei Technologies
A. D'Alessandro
Telecom Italia
October 27, 2014

Dual-Homing Protection for MPLS and MPLS-TP Pseudowires
draft-cheng-pwe3-mpls-tp-dual-homing-protection-01

Abstract

This document describes a framework and several scenarios for pseudowire (PW) dual-homing local protection. A Dual-Node Interconncetion (DNI) PW is provisioned between the dual-homing Provider Edge (PE) nodes for carrying traffic when failure accurs in the Attachment Circuit (AC) or PW side. In order for the dual-homing PE nodes to determine the forwarding state of AC, PW and the DNI PW, necessary state exchange and coordination between the dual-homing PEs are needed. The PW dual-homing local protection mechanism is complementary to the existing PW protection mechanisms.

Requirements Language

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

Status of This Memo

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

Table of Contents

1. Introduction

[RFC6372] and [RFC6378] describe the framework and mechanism of MPLS-TP Linear protection, which can provide protection for the MPLS LSP or PW between the edge nodes. Such mechanism does not protect the failure of the Attachement Circuit (AC) or the Provider Edge (PE) node. [RFC6718] and [RFC6870] describe the framework and mechanism for PW redundancy to provide protection for AC or PE node failure. The PW redundancy mechanism is based on the signaling of Label Distribution Protocol (LDP), which is applicable to PWs with a dynamic control plane. [I-D.ietf-pwe3-endpoint-fast-protection] describes a fast local repair mechanism for PW egress endpoint failures, which is based on PW redundancy, upstream label assignment and context specific label switching. Such mechanism is applicable to PWs with a dynamic control plane.

In some scenarios such as mobile backhauling, the MPLS PWs are provisioned with dual-homing topology, in which at least the CE node in one side is dual-homed to two PEs. If some fault occurs in the primary AC, operators usually prefer to have the switchover only in the dual-homing PE side and keeps the working pseudowires unchanged if possible. This is to avoid massive PWs switchover in the mobile backhaul network due to one AC failure in the core site, and also could achieve efficient and balanced link bandwidth utilization. Similarly, it is preferable to keep using the working AC when one working PW fails in the PSN network. To meet the above requirement, a fast dual-homing PW local protection mechanism is needed to protect the failures in AC, the PE node and the PSN network.

This document describes a framework and several scenarios for pseudowire (PW) dual-homing local protection. A Dual-Node Interconncetion (DNI) PW is provisioned between the dual-homing Provider Edge (PE) nodes for carrying traffic when failure accurs in the AC or PW side. In order for the dual-homing PE nodes to determine the forwarding state of AC, PW and DNI PW, necessary state exchange and coordination between the dual-homing PEs is needed. The mechanism defined in this document is complementary to the existing protection mechanisms. The neccessary protocol extensions will be described in a seperate document.

The proposed mechanism has been deployed in several mobile backhaul networks which use static MPLS-TP PWs for the backhauling of mobile traffic.

2. Reference Models of Dual-homing Local Protection

This section shows the reference architecture of the PE for dual-homing PW local protection and the usage of the architecture in different scenarios.

2.1. PE Architecture

Figure 1 shows the PE architecture for dual-homing local protection. This is based on the architecture in Figure 4a of [RFC3985]. In addition to the AC and the service PW, a DNI PW is provisioned to connect the forwarders of the dual-homing PEs. It can be used to forward traffic between the dual-homing PEs when failure accurs in the AC or service PW side. As [RFC3985] specifies: "any required switching functionality is the responsibility of a forwarder function", in this case, the forwarder is responsible for switching the payloads between three entities: the AC, the service PW and the DNI PW. The specific behavior of forwarder is determined according to the forwarding state machine defined in this document.

```
         +----------------------------------------+
         |           Dual-homing PE Device        |
  Single +----------------------------------------+
    AC   |                 |                      | Service PW
 <------>o    Forwarder    +        Service       X<===========>
         |                 |           PW         |
         +--------+--------+                      |
         |     DNI PW      |                      |
         +--------X--------+----------------------+
                  ^
                  |  DNI PW
                  |
                  V
         +--------X-------------------------------+
         |       Peer Dual-homing PE Device       |
         +----------------------------------------+
```

Figure 1: PE Architecture for Dual-homing Protection

2.2. Dual-Homing Local Protection Reference Scenarios

2.2.1. One-Side Dual-Homing Protection

Figure 2 illustrates the network scenario of dual-homing PW local protection where one of the CEs is dual-homed to two PE nodes. CE1 is dual-homed to PE1 and PE2, while CE2 is single-homed to PE3. DNI-PW is established between the dual-homing PEs, which is used to bridge traffic when a failure occurs in the PSN network or in the AC side. A control mechanism enables the PEs and CE to determine which AC should be used to carry traffic between CE1 and the PSN network. These mechanisms/protocols are beyond the scope of this document. The working and protection PWs can be determined either by configuration or by existing signaling mechanisms.

This scenario can protect the node failure of PE1 or PE2, or the failure of one of the ACs between CE1 and the dual-homing PEs. In addition, dual-homing PW protection can protect the failure occured in the PSN network which impacts the working PW, thus it can be an alternative to PSN tunnel protection mechanisms. This topology can be used in mobile backhauling application scenarios. For example, the NodeB serves as CE2 while the RNC serves as CE1. PE3 works as an access side MPLS device while PE1 and PE2 works as core side MPLS devices.

```
           |<--------------- Emulated Service --------------->|
           |                                                  |
           |          |<------- Pseudo Wire ------>|          |
           |          |                            |          |
           |          |    |<-- PSN Tunnels-->|    |          |
           |          V    V                  V    V          |
           V    AC1   +----+                  +----+          V
     +-----+    |     | PE1|                  |    |          +-----+
     |     |----------|........PW1.(working).......|          |     |
     |     |          |    |                  |    |          |     |
     |     |          +-+--+                  |    |     AC3  |     |
     |     |            |                     |    |      |   |     |
     | CE1 |     DNI PW |                     |PE3 |----------| CE2 |
     |     |            |                     |    |          |     |
     |     |          +-+--+                  |    |          |     |
     |     |          |    |                  |    |          |     |
     |     |----------|......PW2.(protection)......|          |     |
     +-----+    |     | PE2|                  |    |          +-----+
                AC2   +----+                  +----+
```

Figure 2. One-side dual-homing PW protection

Consider in normal state AC1 from CE1 to PE1 is initially active and AC2 from CE1 to PE2 is initially standby, PW1 is the working PW and PW2 is the protection PW.

When a failure occurs in AC1, then the state of AC2 changes to active based on some AC redundancy mechanism. In order to keep the switchover local and continue using PW1 to forward traffic, the forwarder on PE2 needs to connect AC2 to the DNI PW, and the forwarder on PE1 needs to connect the DNI PW to the PW1. In this way the failure in the AC1 do not impact the forwarding of the service PWs across the network. After the switchover, traffic will go through the path: CE1-(AC2)-PE2-(DNI-PW)-PE1-(PW1)-PE3-(AC3)-CE2.

When a failure in the PSN network affects the working PW (PW1), according to PW protection mechanisms, traffic is switched onto the protection PW (PW2), while the state of AC1 remains active. Then the forwarder on PE1 needs to connect AC1 to the DNI PW, and the forwarder on PE2 needs to connect the DNI PW to PW2. In this way the failure in the PSN network do not impact the state of the ACs. After the switchover, traffic will go through the path: CE1-(AC1)-PE1-(DNI-PW)-PE2-(PW2)-PE3-(AC3)-CE2.

In both AC and PW failure cases, the dual-homing PW protection needs to coordinate the PEs to set the forwarding state between the AC, service PW and DNI PW properly.

2.2.2. Two-side Dual-Homing Protection

Figure 3 illustrates the network scenario of dual-homing PW protection where the CEs in both sides are dual-homed. CE1 is dual-homed to PE1 and PE2, and CE2 is dual-homed to PE3 and PE4. A dual-homing control mechanism enables the PEs and CEs to determine which AC should be used to carry traffic between CE and the PSN network. The DNI-PWs are provisioned between the dual-homing PEs on both side. One service PW is established between PE1 and PE3, another service PW is established between PE2 and PE4. The role of working and protection PW can be determined either by configuration or via existing signaling mechansims.

This scenario can protect the node failure of one of the dual-homing PEs, or the failure of one of the ACs between the CEs and their dual-homing PEs. Meanwhile, dual-homing PW protection can protect the failure occured in the PSN network which impacts one of the PWs, thus it can be an alternative to PSN tunnel protection mechanisms. This scenario is mainly used for services provisioning for important business customers. In this case, CE1 and CE2 can be regarded as service access points.

```
           |<---------------- Emulated Service -------------->|
           |                                                  |
           |          |<-------- Pseudowire ------>|          |
           |          |                            |          |
           |          |    |<-- PSN Tunnels-->|    |          |
           |          V    V                  V    V          |
           V    AC1   +----+                  +----+     AC3  V
     +-----+    |     | ...|...PW1.(working)..|... |     |    +-----+
     |     |----------| PE1|                  | PE3|----------|     |
     |     |          +----+                  +----+          |     |
     |     |            |                        |            |     |
     | CE1 |    DNI PW1 |                        |  DNI PW2   | CE2 |
     |     |            |                        |            |     |
     |     |          +----+                  +----+          |     |
     |     |          |    |                  |    |          |     |
     |     |----------| PE2|                  | PE4|--------- |     |
     +-----+    |     | ...|.PW2.(protection).|... |     |    +-----+
                AC2   +----+                  +----+     AC4
```

Figure 3. Two-side dual-homing PW protection

Consider in normal state AC1 from CE1 to PE1 is initially active and AC2 from CE1 to PE2 is initially standby, AC3 from CE2 to PE3 is initially active and AC4 from CE2 to PE4 is initially standby, PW1 is the working PW and PW2 is the protection PW.

When a failure occurs in AC1, the state of AC2 changes to active based on some AC redundancy mechanism. In order to keep the switchover local and continue using PW1 to forward traffic, the forwarder on PE2 needs to connect AC2 to the DNI PW, and the forwarder on PE1 needs to connect the DNI PW with PW1. In this way failures in the AC side do not impact the forwarding of the service PWs across the network. After the switchover, traffic will go through the path: CE1-(AC2)-PE2-(DNI-PW1)-PE1-(PW1)-PE3-(AC3)-CE2.

When a failure occurs in the working PW (PW1), according to the PW protection mechanism, traffic is switched onto the protection PW "PW2". In order to keep the state of AC1 and AC3 unchanged, the forwarder on PE1 needs to connect AC1 to the DNI-PW1, and the forwarder on PE2 needs to connect the DNI-PW1 to PW2. On the other side, the forwarder of PE3 needs to connect AC3 to the DNI-PW2, and the forwarder on PE4 needs to connect PW2 to the DNI-PW2. In this way, the state of the ACs will not be impacted by the failure in the PSN network. After the switchover, traffic will go through the path: CE1-(AC1)-PE1-(DNI-PW1)-PE2-(PW2)-PE4-(DNI-PW2)-PE3-(AC3)-CE2.

In both AC and PW failure cases, the dual-homing PW protection needs to coordinate the PEs to set the forwarding state between the AC, service PW and the DNI PW properly.

3. Generic Dual-homing PW Protection Mechanism

As shown in the above scenarios, with the described Dual-Homing PW Protection, the failures in the AC side do not impact the forwarding behavior of the PWs in the PSN network, and vice-versa. This is achieved by properly setting the forwarding state between the following entities:

- AC

- Service PWs

- DNI PW

The forwarding behavior of the dual-homing PE nodes are determined by the forwarding state machine as shown in table 1:

Service PW	AC	DNI PW	Forwarding Behavior
Active	Active	Up	Service PW <-> AC
Active	Standby	Up	Service PW <-> DNI PW
Standby	Active	Up	DNI PW <-> AC
Standby	Standby	Up	Drop all packets

Table 1. Dual-homing PE Forwarding State Machine

In order for the dual-homing PEs to coordinate the traffic forwarding during the failures, synchronization of the status information of the involved entities and coordination of switchover between the dual-homing PEs are needed. For PWs with a dynamic control plane, such information sychronization and coordination can be achieved with a dynamic protocol, such as [RFC7275], possibly with some extensions. For PWs which are manually configured without a control plane, a new mechanism is needed to exchange the status information and coordinate switchover between the dual-homing PEs. This is described in a separate document.

4. IANA Considerations

This document does not require any IANA action.

5. Security Considerations

The mechanism defined in this document do not affect the security model as defined in [RFC3985].

6. References

6.1. Normative References

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

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

6.2. Informative References

[I-D.ietf-pwe3-endpoint-fast-protection]
Shen, Y., Aggarwal, R., Henderickx, W., and Y. Jiang, "PW Endpoint Fast Failure Protection", draft-ietf-pwe3-endpoint-fast-protection-01 (work in progress), July 2014.

[RFC6372] Sprecher, N. and A. Farrel, "MPLS Transport Profile (MPLS-TP) Survivability Framework", RFC 6372, September 2011.

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

[RFC6718] Muley, P., Aissaoui, M., and M. Bocci, "Pseudowire Redundancy", RFC 6718, August 2012.

[RFC6870] Muley, P. and M. Aissaoui, "Pseudowire Preferential Forwarding Status Bit", RFC 6870, February 2013.

[RFC7275] Martini, L., Salam, S., Sajassi, A., Bocci, M., Matsushima, S., and T. Nadeau, "Inter-Chassis Communication Protocol for Layer 2 Virtual Private Network (L2VPN) Provider Edge (PE) Redundancy", RFC 7275, June 2014.

Authors' Addresses

Weiqiang Cheng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: chengweiqiang@chinamobile.com

Lei Wang
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: Wangleiyj@chinamobile.com

Han Li
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: Lihan@chinamobile.com

Kai Liu
Huawei Technologies
Huawei Base, Bantian, Longgang District
Shenzhen 518129
China

Email: alex.liukai@huawei.com

Shahram Davari
Broadcom Corporation
3151 Zanker Road
San Jose 95134-1933
United States

Email: davari@broadcom.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Alessandro D'Alessandro
Telecom Italia
via Reiss Romoli, 274
Torino 10148
Italy

Email: alessandro.dalessandro@telecomitalia.it

PWE3
Internet-Draft
Updates: 5085 (if approved)
Intended status: Standards Track
Expires: March 6, 2015
T. Nadeau
lucidvision
L. Martini
S. Bryant
Cisco Systems
September 2, 2014

A Unified Control Channel for Pseudowires
draft-ietf-pwe3-vccv-for-gal-02

Abstract

This document describes a unified mode of operation for Virtual Circuit Connectivity Verification (VCCV), which provides a control channel that is associated with a pseudowire (PW). VCCV applies to all supported access circuit and transport types currently defined for PWs, as well as those being transported by the MPLS Transport Profile. This new mode is intended to augment those described in RFC5085. It describes new rules requiring this mode to be used as the default/mandatory mode of operation for VCCV. The older VCCV types will remain optional.

Status of This Memo

Copyright Notice

Table of Contents

1. Requirements Language and Terminology

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].

AC Attachment Circuit [RFC3985].

AVP Attribute Value Pair [RFC3931].

CC Control Channel (used as CC Type).

CE Customer Edge.

CV Connectivity Verification (used as CV Type).

CW Control Word [RFC3985].

L2SS L2-Specific Sublayer [RFC3931].

LCCE L2TP Control Connection Endpoint [RFC3931].

OAM	Operation and Maintenance.
PE	Provider Edge.
PSN	Packet Switched Network [RFC3985].
PW	Pseudowire [RFC3985].
PW-ACH	PW Associated Channel Header [RFC4385].
VCCV	Virtual Circuit Connectivity Verification [RFC5085].

2. Introduction

There is a need for fault detection and diagnostic mechanisms that can be used for end-to-end fault detection and diagnostics for a Pseudowire, as a means of determining the PW's true operational state. Operators have indicated in [RFC4377], and [RFC3916] that such a tool is required for PW operation and maintenance. To this end, the IETF's PWE3 Working Group defined the Virtual Circuit Connectivity Verification Protocol (VCCV) in [RFC5085] . Since then a number of interoperability issues have arisen with the protocol as it is defined.

Over time, a variety of VCCV options or "modes" have been created to support legacy hardware, these modes use of the CW in some cases, while in others the CW is not used. The difficulty of operating these different combinations of "modes" have been detailed in an implementation survey conducted by the PWE3 Working Group and documented in [RFC7079]. The implementation survey and the PWE3 Working Group have concluded that operators have difficulty deploying the VCCV OAM protocol due to the number of combinations and options for its use.

In addition to the implementation issues just described, the ITU-T and IETF have set out to enhance MPLS to make it suitable as an optical transport protocol. The requirements for this protocol are defined as the MPLS Transport Profile (MPLS-TP). The requirements for MPLS-TP can be found in [RFC5654]. In order to support VCCV when an MPLS-TP PSN is in use, the GAL-ACH had to be created [RFC5586]. This resulted in yet another mode of VCCV operation.

This document defines two modes of operation of VCCV: 1) with a control word or 2) without a control word, both with a ACH encapsulation making it possible to handle all of the other cases handled by the other modes of VCCV. The modes of operation defined in this document MUST be implemented.

Figure 1 depicts the architecture of a pseudowire as defined in [RFC3985]. It further depicts where the VCCV control channel resides within this architecture, which will be discussed in detail later in this document.

```
              |<-------------- Emulated Service ---------------->|
              |          |<---------- VCCV ---------->|          |
              |          |<------- Pseudowire ------->|          |
              |          |                            |          |
              |          |    |<-- PSN Tunnel -->|    |          |
              |          V    V                  V    V          |
              V    AC    +----+                  +----+     AC   V
        +-----+    |     | PE1|==================| PE2|     |    +-----+
        |     |----------|............PW1.............|----------|     |
        | CE1 |    |     |    |                  |    |     |    | CE2 |
        |     |----------|............PW2.............|----------|     |
        +-----+  ^ |     |    |==================|    |     | ^  +-----+
              ^  |       +----+                  +----+     | |  ^
              |  |   Provider Edge 1         Provider Edge 2  |  |
              |  |                                            |  |
        Customer |                                            | Customer
        Edge 1   |                                            | Edge 2
                 |                                            |
                 |                                            |
           Native service                               Native service
```

               Figure 1: PWE3 VCCV Operation Reference Model

From Figure 1, Customer Edge (CE) routers CE1 and CE2 are attached to the emulated service via Attachment Circuits (AC), and to each of the Provider Edge (PE) routers (PE1 and PE2, respectively). An AC can be a Frame Relay Data Link Connection Identifier (DLCI), an ATM Virtual Path Identifier / Virtual Channel Identifier (VPI/VCI), an Ethernet port, or any other attachment type for which a PW is defined. The PE devices provide pseudowire emulation, enabling the CEs to communicate over the PSN. A pseudowire exists between these PEs traversing the provider network. VCCV provides several means of creating a control channel over the PW, between the PE routers that attach the PW.

Figure 2 depicts how the VCCV control channel is associated with the pseudowire protocol stack.

```
          +-------------+                                +-------------+
          |  Layer2     |                                |  Layer2     |
          |  Emulated   |       < Emulated Service >     |  Emulated   |
          |  Services   |                                |  Services   |
          +-------------+                                +-------------+
          |             |            VCCV/PW             |             |
          |Demultiplexer|       < Control Channel >      |Demultiplexer|
          +-------------+                                +-------------+
          |    PSN      |          < PSN Tunnel >        |    PSN      |
          +-------------+                                +-------------+
          |  Physical   |                                |  Physical   |
          +-----+-------+                                +-----+-------+
                |                                              |
                |             ____     ___       ____          |
                |           _/    \___/   \    _/    \__       |
                |          /               \__/         \_     |
                |         /                               \    |
                +--------|     MPLS/MPLS-TP or IP Network  |---+
                          \                               /
                           \   ___      ___     __      _/
                            \_/   \____/   \___/  \____/
```

Figure 2: PWE3 Protocol Stack Reference Model including the VCCV Control Channel

VCCV messages are encapsulated using the PWE3 encapsulation as described in Section 3 and Section 4, so that they are handled and processed in the same manner (or in some cases, a similar manner) the PW PDUs for which they provide a control channel. These VCCV messages are exchanged only after the capability (the VCCV Control Channel and Connectivity Verification types) and the desire to exchange VCCV traffic has been advertised between the PEs (see Sections 5.3 and 6.3 of [RFC5085]), and VCCV type to use have been chosen.

[EDITOR’S NOTE - Why are we talking about 6.3 which is L2TPv3 related in a text on GAL?]

3. VCCV Control Channel When The Control Word is Used

When the PWE3 Control Word is used to encapsulate pseudowire traffic, the rules described for encapsulating VCCV CC Type 1 as specified in section 9.5.1 of [RFC6073] and section 5.1.1 of [RFC5085] MUST be used. In this case the advertised CC Type is 1, and Associated Channel Types of 21, 07, or 57 are allowed.

4. VCCV Control Channel When The Control Word is Not Used

When the PWE3 Control Word is not used a new CC Type 4 is defined as follows:

```
0 1
                2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             PW LSE                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            GAL LSE                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 1|Version|   Reserved    |  Associated Channel Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                        VCCV Message Body                      ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

EDITOR’s note = when we wrote RFC3985 I seem to remember that TTL=1 was problematic do we want to specify TTL=1 in the text below?

EDITOR’s note = not sure if it should be MUST or SHOULD in the text below.

When the PW is a single segment PW, the TTL field of the PW Label Stack Entry (LSE) SHOULD be set to 1. In the case of multi-segment pseudo-wires, the PW LSE TTL SHOULD be set to the value needed to reach the intended destination PE as described in [RFC6073].

The GAL LSE MUST contain the GAL reserved label as defined in [RFC5586].

As defined in [RFC4385] and [RFC4446] the first nibble of the next field is set to 0001b to indicate an ACH associated with a pseudowire instead of PW data. The Version and the Reserved fields MUST be set to 0, and the Channel Type is set to 0x0021 for IPv4, 0x0057 for IPv6 payloads [RFC5085] or 0x0007 for BFD payloads [RFC5885].

The Associated Channel Type defines how the "VCCV Message Body" field is to be interpreted by the receiver.

5. VCCV Capability Advertisement

The capability advertisement MUST match the c-bit setting that is advertised in the PW FEC element. If the c-bit is set, indicating the use of the control word, type 1 MUST be advertised and type 4 MUST NOT be advertised. If the c-bit is not set, indicating that the control word is not in use, type 4 MUST be advertised, and type 1 MUST NOT be advertised.

A PE supporting Type 4 MAY advertise other CC types as defined in [RFC5085] . If the remote PE also supports Type 4, then Type 4 MUST be used superseding the Capability Advertisement Selection rules of section 7 from [RFC5085] . If a remote PE does not support Type 4, then the rules from section 7 of [RFC5085] apply. If a CW is in use, then Type 4 is not applicable, and therefore the normal capability advertisement selection rules of section 7 from [RFC5085] apply.

6. Manageability Considerations

Editor’s note - this is a placeholder - I am not sure if it sis needed

7. Security Considerations

This document does not by itself raise any new security considerations beyond those described in [RFC5085].

8. IANA Considerations

8.1. VCCV Interface Parameters Sub-TLV

EDITOR’S NOTE ASFAICS this section can be deleted.

The VCCV Interface Parameters Sub-TLV code point is defined in [RFC4446]. IANA has created and will maintain registries for the CC Types and CV Types (bit masks in the VCCV Parameter ID). The CC Type and CV Type new registries (see Sections 8.1.1 and 8.1.2, respectively of[RFC5085]) have been created in the Pseudo Wires Name Spaces, . The allocations must be done using the "IETF Review" policy defined in [RFC5226].

8.2. MPLS VCCV Control Channel (CC) Type 4

IANA is requested to assign a new bit from the MPLS VCCV Control Channel (CC) Types registry in the PWE3-parameters name space in order to identify VCCV type 4. It is recommended that Bit 3 be assigned to this purpose which would have a value of 0x08.

MPLS VCCV Control Channel (CC) Types

Bit (Value)	Description	Reference
Bit X (0x0Y)	Type 4	[This Specification]

9. Acknowledgements

10. References

10.1. Normative References

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

[RFC3931] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[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, February 2006.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, April 2006.

[RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.

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

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

[RFC5885] Nadeau, T. and C. Pignataro, "Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)", RFC 5885, June 2010.

[RFC6073] Martini, L., Metz, C., Nadeau, T., Bocci, M., and M. Aissaoui, "Segmented Pseudowire", RFC 6073, January 2011.

10.2. Informative References

[RFC3916] Xiao, X., McPherson, D., and P. Pate, "Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)", RFC 3916, September 2004.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4377] Nadeau, T., Morrow, M., Swallow, G., Allan, D., and S. Matsushima, "Operations and Management (OAM) Requirements for Multi-Protocol Label Switched (MPLS) Networks", RFC 4377, February 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC7079] Del Regno, N. and A. Malis, "The Pseudowire (PW) and Virtual Circuit Connectivity Verification (VCCV) Implementation Survey Results", RFC 7079, November 2013.

Authors' Addresses

Thomas D. Nadeau
lucidvision

Email: tnadeau@lucidvision.com

Luca Martini
Cisco Systems

Email: lmartini@cisco.com

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

Internet Working Group   Y. Jiang
Internet Draft   Y. Luo
Intended status: Standards Track   Huawei
                                                             E. Mallette
                                                   Bright House Networks
C. Shen   Y. Shen
China Telecom   Juniper Networks
W. Cheng   G. Zhou
China Mobile   China Unicom

Expires: April 2015   October 25, 2014

Multi-chassis PON Protection in MPLS
draft-jiang-pwe3-mc-pon-03.txt

Status of this Memo

Copyright Notice

Abstract

MPLS is being deployed deeper into operator networks, often to or past the access network node. Separately network access nodes such as PON OLTs have evolved to support first-mile access protection, where one or more physical OLTs provide first-mile diversity to the customer edge. Multi-homing support is needed on the MPLS-enabled PON OLT to provide resiliency for provided services. This document describes the multi-chassis PON protection architecture in MPLS and also proposes the ICCP extension to support it.

Table of Contents

1. Conventions used in this document

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

2. Terminology

DSL Digital Subscriber Line

FTTx Fiber-to-the-x (FTTx, x = H for home, P for premises, C for curb)

ICCP Inter-Chassis Communication Protocol

OLT Optical Line Termination

ONU Optical Network Unit

MPLS Multi-Protocol Label Switching

PON Passive Optical Network

RG Redundancy Group

3. Introduction

MPLS is being extended to the edge of operator networks, as is described in the seamless MPLS use cases [SEAMLESS], and the MS-PW with PON access use case [RFC6456]. Combining MPLS with OLT access further facilitates a low cost multi-service convergence.

Tens of millions of FTTx lines have been deployed over the years, with many of those lines being some PON variant. PON provides operators a cost-effective solution for delivering high bandwidth (1Gbps or even 10Gbps) to a dozen or more subscribers simultaneously.

In the past, access technologies such as Passive Optical Network (PON) and Digital Subscriber Line (DSL) are usually used for subscribers, and no redundancy is provided in their deployment.

But with the rapid growth of mobile data traffic, more and more LTE small cells and Wi-Fi hotspots are deployed. PON is considered as a viable low cost backhaul solution for these mobile services. Besides its high bandwidth and scalability, PON further provides synchronization features, e.g., SyncE and IEEE1588 functionality, which can fulfill synchronization needs of mobile backhaul services.

The Broadband Forum specifies reference architecture for mobile backhaul network using MPLS transport in [TR-221] where PON can be the access technology, and is further working on PON-based mobile backhaul network architecture in [SD-331].

Unlike typical residential service where a single or handful of end-users hangs off of a single PON OLT port in a physical optical distribution network, a PON port that supports a dozen LTE small cells or Wi-Fi hotspots could be providing service to hundreds of simultaneous subscribers. Small cell backhaul often demands the economics of a PON first-mile and yet expects first-mile protection commonly available in point-to-point access portfolio.

Some optical layer of protection mechanisms, such as Trunk and Tree protection, are specified in [IEEE-1904.1] to avoid single point of failure in the access. They are called Type B and Type C protection respectively in [G983.1].

Trunk protection architecture is an economical PON resiliency mechanism, where the working OLT and the working link between the working splitter port and the working OLT (i.e., the working trunk fiber) is protected by a redundant protection OLT and a redundant trunk fiber between the protection splitter port and the protection OLT, however it only protects a portion of the optical path from OLT to ONUs. This is different from the more complex and costly Type C protection architecture where there is a working optical distribution network path from the working OLT and a complete protected optical distribution network path from the protection OLT to the ONUs. Figure 1 demonstrates a typical scenario of Trunk protection.

```
                        |                                 |
                        |<--Optical Distribution Network->|
                        |                                 |
                        |   branch               trunk    +-----+
                  +-----+   fibers               fibers   |     |
   Base     ------|     |     |                           . OLT |
   Stations ------| ONU |\    |                        ,''|  A  |
            ------|     |  \  V                     _-'   +-----+
                  +-----+    \                    .'
                            .  \  +----------+ ,-'
                  +-----+   .    \|          -'   Working
   Base     ------|     |   .     | Optical  |
   Stations ------| ONU |---------| Splitter |
            ------|     |   .    /|          -,   Protection
                  +-----+   .  /  +----------+ ''.,
                             /                     '-,    +-----+
                  +-----+  /                          ''.,|     |
   Base     ------|     |/                                | OLT |
   Stations ------| ONU |                                 |  B  |
            ------|     |                                 +-----+
                  +-----+
```

Figure 1 Trunk Protection Architecture in PON

Besides small cell backhaul, this protection architecture can also be applicable to other services, for example, DSL and Multi-System Operator (MSO) services. In that case, an ONU in Figure 1 can play the similar role as a Digital Subscriber Line Access Multiplexer (DSLAM) and dozens of Customer Premises Equipments (CPEs) or cable modems may be attached to it.

In some deployments, it is also possible that only some ONUs are needed to be protected.

The PON architecture depicted in Figure 1 can provide redundancy in its physical topology, however, all traffic including link OAM are blocked on the protection link which frustrates end to end protection mechanisms such as ITU-T G.8031. Therefore, some standard signaling mechanisms are needed between OLTs to exchange information, for example, PON link status, registered ONU information, and network status, so that protection and restoration can be done both rapidly and reliably, especially when the OLTs also support MPLS.

ICCP [ICCP] provides a framework for inter-chassis synchronization of state and configuration data between a set of two or more PEs. Currently ICCP only defines application specific messages for PW redundancy and mLACP, but it can be easily extended to support PON as an Attachment Circuit (AC) redundancy.

This document proposes the extension of ICCP to support Multi-chassis PON protection in MPLS.

4. ICCP Protocol Extensions

4.1. Multi-chassis PON Application TLVs

A set of multi-chassis PON application TLVs are defined in the following sub-sections.

4.1.1. PON Connect TLV

This TLV is included in the RG Connect message to signal the establishment of PON application connection.

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|   Type=0x00XX             |    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Protocol Version         |A|        Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Optional Sub-TLVs                          |
~                                                               ~
|                                                               |
+                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ...                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- U and F Bits, both are set to 0.

- Type, set to 0x00XX for "PON Connect TLV".

- Length, Length of the TLV in octets excluding the U-bit, F-bit, Type, and Length fields.

- Protocol Version, the version of this PON specific protocol for the purposes of inter-chassis communication. This is set to 0x0001.

- A Bit, Acknowledgement Bit. Set to 1 if the sender has received a PON Connect TLV from the recipient. Otherwise, set to 0.

- Reserved, Reserved for future use.

- Optional Sub-TLVs, there are no optional Sub-TLVs defined for this version of the protocol.

4.1.2. PON Disconnect TLV

This TLV is included in the RG Disconnect message to indicate that the connection for the PON application is to be terminated.

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |U|F|   Type=0x00XX             |    Length                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Optional Sub-TLVs                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- U and F Bits, both are set to 0.

- Type, set to 0x00XX for "PON Disconnect TLV".

- Length, Length of the TLV in octets excluding the U-bit, F-bit, Type, and Length fields.

- Optional Sub-TLVs, there are no optional Sub-TLVs defined for this version of the protocol.

4.1.3. PON Configuration TLV

The "PON Configuration TLV" is included in the "RG Application Data" message, and announces an OLT’s system parameters to other members in the same RG.

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |U|F|   Type=0x00XX             |    Length                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          System ID                            |
 |                                                              |
 |                                                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     System Priority           |             Port ID           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- U and F Bits, both are set to 0.

- Type, set to 0x00XX for "PON Configuration TLV".

- Length, Length of the TLV in octets excluding the U-bit, F-bit, Type, and Length fields.

- System ID, 8 octets encoding the System ID used by the OLT, which is the Chassis MAC address. If a 6 octet System ID is used, the least significant 2 octets of the 8 octet field will be encoded as 0000.

- System Priority, 2 octets encoding the System Priority.

- Port ID, 2 octets PON Port ID.

Further configuration considerations such as multicast table and ARP table for static MAC addresses will be added in a next version.

4.1.4.PON State TLV

The "PON State TLV" is included in the "RG Application Data" message, and used by an OLT to report its PON states to other members in the same RG.

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|   Type=0x00XX             |    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ROID                             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Local PON Port state                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Remote PON Port state                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- U and F Bits, both are set to 0.

- Type, set to 0x00XX for "PON State TLV"

- Length, Length of the TLV in octets excluding the U-bit, F-bit, Type, and Length fields.

- ROID, as defined in the ROID section of [ICCP].

- Local PON Port State, the status of the local PON port as determined by the sending OLT (PE). The last bit is defined as Fault indication of the PON Port associated with this PW (1 - in fault).

- Remote PON Port State, the status of the remote PON port as determined by the remote peer of the sending OLT (PE). The last bit is defined as Fault indication of the PON Port associated with this PW (1 - in fault).

4.1.5.PON ONU Database Sync TLV

This TLV is used to communicate the registered ONU database associated with a PON port between the active and standby OLT. This message is used to both transmit the PON ONU Database from working OLT to protect OLT and to communicate the PON ONU database status between protect OLT and working OLT.

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|   Type=0x00XX             |    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             ROID                              |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|  Reserved   |                    OUI                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                      ONU Database Entry1                      |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- U and F Bits, both are set to 0.

- Type, set to 0x00XX for "PON ONU Database Sync TLV"

- Length, Length of the TLV in octets excluding the U-bit, F-bit, Type, and Length fields.

- ROID, defined in the ROID section of [ICCP].

- A bit, Acknowledgement bit. Set to 1 if the receiver has received a PON ONU Database Sync. Otherwise, set to 0.

- Reserved, reserved for future use.

- OUI, the 3-byte [IEEE-802.3] organization unique identifier that uniquely identifies the format for describing the registered ONU database information. There are multiple PON standards and are varying implementations within a given PON standard which likely have different required information, format, etc., related to the ONU Database Entry.

- ONU Database Entry, there may be one or more ONU Database Entries transmitted in the PON ONU Database Sync TLV, each of which would describe a registered ONU. The format of the ONU Database Entry is outside the scope of this document and will be defined by the relevant PON standard organization.

5. PON ONU Database Synchronization

Without an effective mechanism to communicate the registered ONUs between the working and protection OLT, all registered ONUs would be de-registered and go through re-registration during a switchover, which would significantly increase protection time. To enable faster switchover capability, the work OLT must be able to communicate the registered ONUs associated with an ROID to the protection OLT.

The PON ONU Database Synchronization would begin once the ICCP PON Application enters OPERATIONAL state. The working OLT, the one with the working link member for the ROID, would begin transmitting the database of actively registered ONUs to the protection OLT for the same ROID. Each instance of the PON ONU Database Sync TLV describes a set of ONU Database Entries. Each ONU Database Entry would describe a registered ONU.

The transmission of PON ONU Database Descriptors for a given ROID is only unidirectional - from the working OLT to the protection OLT. The protection OLT would only be responsible for acknowledging the received message to provide a reliable database synchronization mechanism. As ONUs register and deregister from the working OLT, the working OLT would transmit PON ONU Database Synchronization TLV including only the updated ONU Database Entries.

If protected ONUs and unprotected ONUs are miscellaneously attached to the same splitter, only the protected ONUs needs to be synchronized. The specific ONUs which needs to be synchronized can be policy driven and provisioned in the management plane, or by some other signaling options.

6. Multi-chassis PON application procedures

Two typical MPLS protection network architectures for PON access are depicted in Fig.2 and Fig.3 (their PON access segments are the same as in Fig.1 and thus omitted for simplification). OLTs with MPLS functionality are connected to a single PE (Fig.2) or dual home PEs (Fig.3) respectively, i.e., the working OLT to PE1 by a working PW and the protection OLT to PE1 or PE2 by a protection PW, thus these devices constitute an MPLS network which provides PW transport services between ONUs and a CE.

```
                  +-----+
                  |     |
                  |OLT  -,
                  | A   | '.,
                  +-----+    ', PW1
                               '',
                                  '.,   +-----+          +-----+
                                     ', |     |          |     |
                                       '. PE1 -----------   CE |
                                     .''|     |          |     |
                                  ,-'   +-----+          +-----+
                               .,'
                  +-----+    .'' PW2
                  |     | ,-'
                  |OLT  -'
                  | B   |
                  +-----+
```

Figure 2 An MPLS Network with a Single PE

```
                  +-----+               +-----+
                  |     |     PW1       |     |
                  |OLT  ----------------- PE1 -,
                  | A   |               |     | ',
                  +-----+               +--/--+   ',
                                           |        ',
                                           |          '. +-----+
                                           |            ''     |
                                           |             |  CE |
                                           |             .     |
                                           |           ,'+-----+
                                           |        ,-'
                  +-----+               +--\--+   ,'
                  |     |     PW2       |     | .'
                  |OLT  ----------------- PE2 -'
                  | B   |               |     |
                  +-----+               +-----+
```

Figure 3 An MPLS Network with Dual-homing PEs

Faults may be encountered in PON access links, or in the MPLS network (including the working OLT). Procedures for these cases are described in this section (it is assumed that both OLTs and PEs are working in independent mode of PW redundancy [RFC6870]).

6.1. Protection procedure upon PON link failures

When a fault is detected on a working PON link, a working OLT MUST turn off its associated PON interface so that the protection trunk link to the protection OLT can be activated, then it MUST send an LDP fault notification message (i.e., with the status bit "Local AC (ingress) Receive Fault " being set) to its peer PE on the remote end of the PW. At the same time, the working OLT MUST send an ICCP message with PON State TLV with local PON Port State being set to notify the protection OLT of the PON fault.

Upon receiving a PON state TLV where Local PON Port state is set, a protection OLT MUST activate the protection PON link in the protection group, and advertise a notification message for the protection PW with the Preferential Forwarding status bit of active to the remote PE.

According to [RFC6870], the remote PE(s) can match the local and remote Preferential Forwarding status and select PW2 as the new active PW to which to send traffic.

6.2. Protection procedure upon PW failures

Usually MPLS networks have its own protection mechanism such as LSP protection or Fast Reroute (FRR). But in a link sparse access or aggregation network where protection for a PW is impossible in its LSP layer, the following PW layer protection procedures can be enabled.

When a fault is detected on its working PW (e.g., by VCCV BFD), a working OLT SHOULD turn off its associated PON interface and then send an ICCP message with PON State TLV with local PON Port State being set to notify the protection OLT of the PON fault.

Upon receiving a PON state TLV where Local PON Port state is set, the protection OLT MUST activate its PON interface to the protection trunk fiber. At the same time, the protection OLT MUST send a notification message for the protection PW with the Preferential Forwarding status bit of active to the remote PE, so that traffic can be switched to the protection PW.

6.3. Protection procedure upon the working OLT failure

As depicted in Fig. 2, a service is provisioned with a working PW and a protection PW, both PW terminated on PE1. If PE1 lost its connection to the working OLT, it SHOULD send a LDP notification message on the protection PW with the Request Switchover bit set.

Upon receiving a LDP notification message from its remote PE with the Request Switchover bit set, a protection OLT MUST activate its optical interface to the protection trunk fiber and activate the associated protection PW, so that traffic can be reliably switched to the protection trunk PON link and the protection PW.

In the case of Fig.3, PW-RED State TLV [ICCP] can be used by PE1 to notify PE2 the faults in all the scenarios, and PE2 operates the same as described in Section 5.1 to 5.3.

7. Security Considerations

Security considerations as described in [ICCP] apply.

8. IANA Considerations

These values are requested from the registry of "ICC RG parameter type":

0x00X0	PON Connect TLV
0x00X1	PON Disconnect TLV
0x00X2	PON Configuration TLV
0x00X3	PON State TLV
0x00X4	PON ONU Database Sync TLV

9. References

9.1. Normative References

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

[RFC6870] Muley, P., Aissaoui, M., "Pseudowire Preferential Forwarding Status Bit", RFC 6870, February 2013

[ICCP] Martini, L. and et al, "Inter-Chassis Communication Protocol for L2VPN PE Redundancy", RFC 7275, June 2014

9.2. Informative References

[RFC6456] Li, H., Zheng, R., and Farrel, A., "Multi-Segment Pseudowires in Passive Optical Networks", RFC 6456, November 2011

[SEAMLESS] Leymann, N., and et al, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-04, Work in progress

[G983.1] ITU-T, "Broadband optical access systems based on Passive Optical Networks (PON)", ITU-T G.983.1, January, 2005

[IEEE-1904.1] IEEE Std. 1904.1, "Standard for Service Interoperability in Ethernet Passive Optical Networks (SIEPON)", IEEE Computer Society, June, 2013

[IEEE-802] IEEE Std. 802, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE Computer Society, December, 2001 with amendments

[TR-221] BBF TR-221, "Technical Specifications for MPLS in Mobile Backhaul Networks", the Broadband Forum, October, 2011

[SD-331] BBF SD-331, "Architecture and Technical Requirements for PON-Based Mobile Backhaul Networks", the Broadband Forum, Work in progress

10. Acknowledgments

The authors would like to thank Min Ye, Hongyu Li, Wei Lin, Xifeng Wan, Yannick Legoff and Shrinivas Joshi for their valuable discussions and comments.

Authors' Addresses

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
Email: jiangyuanlong@huawei.com

Yong Luo
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
Email: dennis.luoyong@huawei.com

Edwin Mallette
Bright House Networks
4145 S. Falkenburg Road
Tampa, FL 33578 USA
Email: edwin.mallette@gmail.com

Chengbin Shen
China Telecom
Email: shencb@sttri.com.cn

Yimin Shen
Juniper Networks
10 Technology Park Drive
Westford, MA 01886, USA
Email: yshen@juniper.net

Weiqiang Cheng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053, China
Email: chengweiqiang@chinamobile.com

Guangtao Zhou
China Unicom
No.9 Shouti South Road
Beijing 100048, China
Email: zhouguangtao@chinaunicom.cn

Network Working Group A. Malis
Internet-Draft L. Andersson
Updates: 6870 (if approved) Huawei Technologies Co., Ltd
Intended status: Standards Track H. van Helvoort
Expires: April 13, 2015 Hai Gaoming BV
J. Shin
SK Telecom
L. Wang
China Mobile
A. D'Alessandro
Telecom Italia
October 10, 2014

S-PE Outage Protection for Static Multi-Segment Pseudowires
draft-shawam-pwe3-ms-pw-protection-02.txt

Abstract

In MPLS and MPLS-TP environments, statically provisioned Single-Segment Pseudowires (SS-PWs) are protected against tunnel failure via MPLS-level and MPLS-TP-level tunnel protection. With statically provisioned Multi-Segment Pseudowires (MS-PWs), each segment of the MS-PW is likewise protected from tunnel failures via MPLS-level and MPLS-TP-level tunnel protection. However, static MS-PWs are not protected end-to-end against failure of one of the switching PEs (S-PEs) along the path of the MS-PW. This document describes how to achieve this protection by updating the existing procedures in RFC 6870. It also contains an optional approach based on MPLS-TP Linear Protection.

Status of This Memo

Table of Contents

1. Introduction

As described in RFC 5659 [RFC5659], Multi-Segment Pseudowires (MS-PWs) consist of terminating PEs (T-PEs), switching PEs (S-PEs), and PW segments between the T-PEs at each of the MS-PW and the interior S-PEs. In MPLS and MPLS-TP environments, statically provisioned Single-Segment Pseudowires (SS-PWs) are protected against tunnel failure via MPLS-level and MPLS-TP-level tunnel protection. With statically provisioned Multi-Segment Pseudowires (MS-PWs), each PW segment of the MS-PW is likewise protected from tunnel failure via MPLS-level and MPLS-TP-level tunnel protection. However, PSN tunnel protection does not protect static MS-PWs from failures of S-PEs along the path of the MS-PW.

RFC 6718 [RFC6718] provides a general framework for PW protection, and RFC 6870 [RFC6870], which is based upon that framework, describes protection procedures for MS-PWs that are dynamically signaled using LDP. This document describes how to achieve protection against S-PE failure in a static MS-PW by extending RFC 6870 to be applicable for statically provisioned MS-PWs pseudowires (PWs) as well.

This document also contains an optional alternative approach based on MPLS-TP Linear Protection. This approach, described in Appendix A, MUST be identically provisioned in the PE endpoints for the protected MS-PW in order to be used. See Appendix A for further details on this alternative approach.

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. Extension to RFC 6870 to Protect Statically Provisioned SS-PWs and MS-PWs

Section 3.2.3 of RFC 6718 and Section A.5 of RFC 6870 document how to use redundant MS-PWs to protect an MS-PW against S-PE failure in the case of a singly-homed CE, using the following network model from RFC 6718:

```
       Native   |<----------- Pseudowires ----------->|  Native
       Service  |                                     |  Service
        (AC)    |     |<-PSN1-->|     |<-PSN2-->|     |  (AC)
          |     V     V         V     V         V     V   |
          |     +-----+         +-----+         +-----+   |
   +----+ |     |T-PE1|=========|S-PE1|=========|T-PE2|   |   +----+
   |    |-------|......PW1-Seg1.......|.PW1-Seg2......|-------|    |
   | CE1|       |     |=========|     |=========|     |       | CE2|
   |    |       +-----+         +-----+         +-----+       |    |
   +----+        |.||.|                          |.||.|       +----+
                 |.||.|         +-----+          |.||.|
                 |.||.|=========|     |========== .||.|
                 |.||...PW2-Seg1......|.PW2-Seg2...||.|
                 |.| ===========|S-PE2|============ |.|
                 |.|            +-----+             |.|
                 |.|============+-----+============= .|
                 |.....PW3-Seg1.|     | PW3-Seg2......|
                  ==============|S-PE3|===============
                                |     |
                                +-----+
```

Figure 1: Single-Homed CE with Redundant MS-PWs

In this figure, CE1 is connected to PE1 and CE2 is connected to PE2. There are three MS PWs. PW1 is switched at S-PE1, PW2 is switched at S-PE2, and PW3 is switched at S-PE3. This scenario provides N:1 protection against S-PE failure for the subset of the path of the emulated service from T-PE1 to T-PE2.

The procedures in RFCs 6718 and 6870 rely on LDP-based PW status signaling to signal the state of the primary MS-PW that is being protected, and the precedence in which redundant MS-PW(s) should be used to protect the primary MS-PW should it fail. These procedures make use of information carried by the PW Status TLV, which for dynamically signaled PWs is carried by the LDP protocol.

However, statically provisioned PWs (SS-PWs or MS-PWs) do not use the LDP protocol for PW set and signaling, rather they are provisioned by network management systems or other means at each T-PE and S-PE along their path. They also do not use the LDP protocol for status signaling. Rather, they use procedures defined in RFC 6478 [RFC6478] for status signaling via the PW OAM message using the PW Associated Channel Header (ACH). The PW Status TLV carried via this status signaling is itself identical to the PW Status TLV carried via LDP-based status signaling, including the identical PW Status Codes.

Sections 6 and 7 of RFC 6870 describes the management of a primary PW and its secondary PW(s) to provide resiliency to the failure of the primary PW. They use status codes transmitted between endpoint T-PEs using the PW Status TLV transmitted by LDP. For this management to apply to statically provisioned PWs, the PW status signaling defined in RFC 6478 MUST be used for the primary and secondary PWs. In that case, the endpoint T-PEs can then use the PW status signaling provided by RFC 6478 in the place of LDP-based status signaling, but otherwise operate identically as described in RFC 6870.

3. Operational Considerations

Because LDP is not used between the T-PEs for statically provisioned MS-PWs, the negotiation procedures described in RFC 6870 cannot be used. Thus, operational care must be taken so that the endpoint T-PEs are identically provisioned regarding the use of this document, specifically whether or not MS-PW redundancy is being used, and for each protected MS-PW, the identity of the primary MS-PW and the precedence of the secondary MS-PWs.

4. Security Considerations

The security considerations defined for RFC 6478 apply to this document as well. As the security considerations in RFCs 6718 and 6870 are related to their use of LDP, they are not required for this document.

If the alternative approach in Appendix A is used, then the security considerations defined for RFCs 6378, 7271, and 7324 also apply.

5. IANA Considerations

There are no requests for IANA actions in this document.

Note to the RFC Editor - this section can be removed before publication.

6. Acknowledgements

The authors would like to thank Matthew Bocci, Yaakov Stein, and David Sinicrope for their comments on this document.

Figure 1 and the explanatory paragraph following the figure were taken from RFC 6718.

7. References

7.1. Normative References

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

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

[RFC6478] Martini, L., Swallow, G., Heron, G., and M. Bocci, "Pseudowire Status for Static Pseudowires", RFC 6478, May 2012.

[RFC6870] Muley, P. and M. Aissaoui, "Pseudowire Preferential Forwarding Status Bit", RFC 6870, February 2013.

[RFC7271] Ryoo, J., Gray, E., van Helvoort, H., D'Alessandro, A., Cheung, T., and E. Osborne, "MPLS Transport Profile (MPLS-TP) Linear Protection to Match the Operational Expectations of Synchronous Digital Hierarchy, Optical Transport Network, and Ethernet Transport Network Operators", RFC 7271, June 2014.

[RFC7324] Osborne, E., "Updates to MPLS Transport Profile Linear Protection", RFC 7324, July 2014.

7.2. Informative References

[RFC5659] Bocci, M. and S. Bryant, "An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge", RFC 5659, October 2009.

[RFC6718] Muley, P., Aissaoui, M., and M. Bocci, "Pseudowire Redundancy", RFC 6718, August 2012.

Appendix A. Optional Linear Protection Approach

A.1. Introduction

In "MPLS Transport Profile (MPLS-TP) Linear Protection" [RFC6378], as well as in the later updates of this RFC in "MPLS Transport Profile (MPLS-TP) Linear Protection to Match the Operational Expectations of SDH, OTN and Ethernet Transport Network Operators" [RFC7271] and in "Updates to MPLS Transport Profile Linear Protection" [RFC7324], the Protection State Coordination (PSC) protocol was defined for MPLS LSPs only.

This Appendix extends these RFCs to be applicable for PWs (SS-PW and MS-PW) as well. This is useful especially in the case of end-to-end static provisioned MS-PWs running over MPLS-TP where tunnel protection alone cannot be relied upon for end-to-end protection of PWs against S-PE failure. It also enables a uniform operational approach for protection at LSP and PW layers and an easier management integration for networks that already use RFCs 6378, 7271, and 7324.

This Appendix is optional alternative approach to the one in Section 2, therefore all implementations MUST include the approach in Section 2 even if this alternative approach is used. The operational considerations in Section 3 continue to apply when this approach is used, and operational care must be taken so that the endpoint T-PEs are identically provisioned regarding the use of this document.

A.2. Encapsulation of the PSC Protocol for Pseudowires

The PSC protocol can be used to protect against defects on any LSP (segment, link or path). In the case of MS-PW, the PSC protocol can also protect failed intermediate nodes (S-PE). Linear protection protects an LSP or PW end-to-end and if a failure is detected, switches traffic over to another (redundant) set of resources.

Obviously, the protected entity does not need to be of the same type as the protecting. For example, it is possible to protect a link by a path. Likewise it is possible to protect a SS-PW with a MS-PW and vice versa.

From a PSC protocol point of view it is possible to view a SS-PW as a single hop LSP, and a MS-PW as a multiple hop LSP. Thus, this provides end-to-end protection for the SS-PW or MS-PW. The G-ACh carrying the PSC protocol information is placed in the label stack directly beneath the PW identifier. The PSC protocol will then work as specified in RFCs 6378, 7271, and 7324.

Authors' Addresses

Andrew G. Malis
Huawei Technologies Co., Ltd

Email: agmalis@gmail.com

Loa Andersson
Huawei Technologies Co., Ltd

Email: loa@mail01.huawei.com

Huub van Helvoort
Hai Gaoming BV

Email: huubatwork@gmail.com

Jongyoon Shin
SK Telecom

Email: jongyoon.shin@sk.com

Lei Wang
China Mobile

Email: wangleiyj@chinamobile.com

Alessandro D'Alessandro
Telecom Italia

Email: alessandro.dalessandro@telecomitalia.it

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 25, 2015

S. Zhuang
H. Wang
Z. Li
Huawei Technologies

August 24, 2014

Yang Model for L2VPN
draft-zhuang-l2vpn-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage L2VPN. Both VPWS and VPLS are supported.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

This document defines a YANG data model that can be used to configure and manage L2VPN. Both VPWS and VPLS are supported.

2. Terminology

L2VPN: Layer 2 Virtual Private Network

VPLS: Virtual Private LAN Service

VPWS: Virtual Private Wire Service

3. Design of Data Model

3.1. Overview

The L2VPN Yang module is divided in following containers :

o l2vpncommon : that contains common writable configuration and readable objects for VPWS and VPLS.

o l2vpnvpws : that contains writable configuration and readable objects for VPWS.

o l2vpnvpls: that contains writable configuration and readable objects for VPLS.

The figure below describe the overall structure of the L2VPN Yang module :

```
module: l2vpn
   +--rw l2vpncommon
   ...
   +--rw l2vpnvpws
   ...
   +--rw l2vpnvpls
   ...
      ...
```

3.2. L2VPN Common Configuration

L2VPN common configuration container includes the global parameters for L2VPN, PW template configuration, etc. These parameters can be used by both VPWS and VPLS.

PW template configuration includes peer address, control word, MTU, sequence number, tunnel policy, parameters of AC, etc.

```
+--rw l2vpncommon
|  +--rw l2vpnGlobal
|  |  +--rw l2vpnEnable             boolean
|  |  +--rw vplsLoopDetectEnable?   boolean
|  +--rw pwTemplates
|  |  +--rw pwTemplate* [pwTemplateName]
|  |     +--rw pwTemplateName          string
|  |     +--rw peerAddr?               inet:ip-address
|  |     +--rw mtu?                    uint16
|  |     +--rw ctrlWord?               enumeration
|  |     +--rw tunnelPolicy?           string
|  |     +--rw tdmEncapsulateNumber?   uint8
|  |     +--rw jitterBuffer?           uint16
|  |     +--rw rtpHeader?              boolean
|  |     +--rw idleCode?               string
|  |     +--rw tdmSequenceNumber?      boolean
|  |     +--rw payloadCompression?     boolean
|  |     +--rw timeSlot?               uint8
|  |     +--rw maxAtmCells?            uint8
|  |     +--rw atmPackOvertime?        uint16
|  |     +--rw atmTransmitCell?        uint8
|  |     +--rw sequenceNumber?         boolean
...
```

3.3. VPWS Configuration

The VPWS configuration container includes VPWS instance configuration, VPWS switch instance configuration and VPWS statistics information.

```
+--rw l2vpnvpws
|  +--rw vpwsStatisticInfo
|  ...
|  +--rw vpwsInstances
|  ...
|  +--rw vpwsSwitchInstances
|  ...
```

3.3.1. VPWS Instances Configuration

The VPWS instance configuration includes per-instance parameters, AC configuration, PW configuration, TDM parameters, ATM parameters, reliability (including PW redundancy) configuration etc.

```
+--rw vpwsInstances
|  +--rw vpwsInstance* [instanceName instanceType]
|     +--rw instanceName       leafref
|     +--rw instanceType       instanceType
|     +--rw encapsulateType?   pw-encapsulation
|     +--rw description?       string
|     +--ro instanceState?     enumeration
|     +--ro lastUpTime?        yang:date-and-time
|     +--ro totalUpTime?       string
|     +--rw tdmParameters
|     |  +--rw tdmEncapsulateNumber?   uint8
|     |  ...
|     +--rw atmParameters
|     |  ...
|     +--rw l2vpnAcs
|     |  ...
|     +--rw vpwsPws
|     |  ...
|     +--rw reliabilitys
|        +--rw reliability* [pwRedundancyMode]
...
```

3.3.2. VPWS Switch Instances Configuration

VPWS switch instance configuration includes the configuration for multi-segment PW such as per-instance parameters, PW configuration, ATM parameters, TDM parameters etc.

```
   +--rw vpwsSwitchInstances
      +--rw vpwsSwitchInstance* [instanceName instanceType]
         +--rw instanceName       string
         +--rw instanceType       instanceType
         +--rw encapsulateType?   pw-encapsulation
         +--rw switchType?        enumeration
         +--rw ctrlWordTrans?     boolean
         +--rw controlWord?       enumeration
         +--ro instanceState?     enumeration
         +--ro createTime?        string
         +--ro upTime?            string
         +--ro lastChgTime?       string
         +--ro lastUpTime?        yang:date-and-time
         +--ro totalUpTime?       string
         +--rw vpwsPws
            +--rw vpwsPw* [pwRole pwId]
               +--rw pwRole            pw-role
               +--rw pwId              uint32
               +--rw peerIp?           inet:ip-address
               +--rw transmitLabel?    uint32
               +--rw receiveLabel?     uint32
               +--rw ctrlWord?         enumeration
               +--rw vccvAbility?      boolean
               +--rw tnlPolicyName?    string
               +--rw pwTemplateName?   string
               +--rw requestVlanId?    uint16
               +--rw vlanTpId?         string
               +--rw pwTtl?            uint8
               +--rw tdmParameters
               ...
               +--rw atmParameters
               ...
               +--rw vpwsLdpPwInfo
   ...
```

3.3.3. VPWS Statistics Information

The VPWS statistics information container includes statistics information of VPWS.

```
+--rw vpwsStatisticInfo
|  +--rw vpwsLdpAcStatInfo
|  |  +--ro totalLdpAcNum?   uint32
|  |  +--ro upLdpAcNum?      uint32
|  |  +--ro downLdpAcNum?    uint32
|  +--rw vpwsLdpPwStatInfo
|  |  +--ro totalLdpPwNum?   uint32
|  |  +--ro upLdpPwNum?      uint32
|  |  +--ro downLdpPwNum?    uint32
|  +--rw vpwsLdpPwRemoteStatInfo
|  |  +--ro remoteVcNum?   uint32
|  +--rw vpwsSwitchInstanceStatInfo
|     +--ro totalSwitchInstanceNum?   uint32
|     +--ro upSwitchInstanceNum?      uint32
|     +--ro downSwitchInstanceNum?    uint32
```

3.4. VPLS Configuration

The L2VPN VPLS configuration includes VPLS instance configuration, VPLS statistics information.

```
+--rw l2vpnvpls
   +--rw vplsStatisticInfo
   |  +--rw vplsInstStatisticsInfo
   ...
   |  +--rw vplsPwStatisticsInfo
   ...
   |  +--rw vplsAcStatisticsInfo
   ...
   |  +--ro vplsTnlRefInfos
   ...
   |  +--rw vplsLoopDetectStaticInfo
   |     +--ro totalVplsLoopDetectNum?   uint32
   +--rw vplsInstances
      +--rw vplsInstance* [instanceName]
         +--rw instanceName            string
         +--rw description?            string
         +--rw memberDiscoveryMode?    enumeration
         +--rw encapsulateType?        pw-encapsulation
         +--rw mtuValue?               uint16
         ...
```

3.4.1. VPLS Instance Configuration

The VPLS instance configuration includes member discovery mode, encapsulate type, VPLS LDP instance configuration, VPLS BGP AD instance configuration, VPLS BGP instance configuration and VPLS ACs configuration etc.

-- VPLS LDP instance configuration: This configuration describes how to configure LDP-based VPLS, with the signaling type being LDP.

-- VPLS BGP AD instance configuration: This configuration describes how to configure BGP AD VPLS to exchange extended BGP packets to automatically discover member VSIs in a VPLS domain and then use LDP FEC 129 to negotiate PW establishment to achieve automatic VPLS PW deployment.

-- VPLS BGP instance configuration: This configuration describes how to configure BGP VPLS. Detailed operations include configuring BGP as the signaling protocol, and configuring VPN targets to implement automatic discovery of VPLS PEs.

-- VPLS ACs configuration: This configuration describes configuration parameters of ACs.

```
+--rw vplsInstances
   +--rw vplsInstance* [instanceName]
      +--rw instanceName            string
      +--rw description?            string
      +--rw memberDiscoveryMode?    enumeration
      +--rw encapsulateType?        pw-encapsulation
      +--rw mtuValue?               uint16
      ...
      +--rw vsiPipe
      ...
      +--rw vplsLdpInst
      |  +--rw vsiId?               uint32
      ...
      +--rw vplsBgpAdInst
      |  +--rw vplsId?           string
      |  +--ro bgpAdRd?          string
      |  +--ro vsiId?            inet:ip-address
      |  +--rw vpnTargets
      ...
      +--rw vplsBgpInst
      |  +--rw bgpRd?          string
      |  +--rw ignoreMtu?      boolean
      ...
      +--rw vplsAcs
      |  +--rw vplsAc* [interfaceName]
      ...
      +--rw vplsLoopDetectInfo
         ...
```

3.4.2. VPLS Statistics Information

The VPLS statistics information container includes VPLS instance statistics information, VPLS PW statistics information, VPLS AC statistics information etc.

```
+--rw vplsStatisticInfo
|  +--rw vplsInstStatisticsInfo
...
|  +--rw vplsPwStatisticsInfo
...
|  +--rw vplsAcStatisticsInfo
...
|  +--ro vplsTnlRefInfos
...
|  +--rw vplsLoopDetectStaticInfo
|     +--ro totalVplsLoopDetectNum?   uint32
...
```

4. L2VPN Yang Module

```
<CODE BEGINS> file "l2vpn-yang@2014-08-21.yang"

module l2vpn {
    namespace "urn:huawei:params:xml:ns:yang:l2vpn";
    prefix "l2vpn";

    /* import */
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-yang-types {
        prefix yang;
    }
    description
        "This YANG module defines the generic configuration data for
         L2VPN services.

         Terms and Acronyms
         L2VPN: Layer 2 Virtual Private Network
         VPLS: Virtual Private LAN Service
         VPWS: Virtual Private Wire Service
         ...
         ";
```

```
    revision 2014-08-21 {
        description
            "Initial revision.";
    }

    /* Typedef */
    typedef pw-encapsulation {
        description "PW encapsulation type.";
        type enumeration {
            enum fr {
                value "0";
                description "fr:";
            }
            enum atm-aal5-sdu {
                value "1";
                description "atm-aal5-sdu:";
            }
            enum atm-trans-cell {
                value "2";
                description "atm-trans-cell:";
            }
            enum vlan {
                value "3";
                description "vlan:";
            }
            enum ethernet {
                value "4";
                description "ethernet:";
            }
            enum hdlc {
                value "5";
                description "hdlc:";
            }
            enum ppp {
                value "6";
                description "ppp:";
            }
            enum cem {
                value "7";
                description "cem:";
            }
            enum atm-nto1-vcc {
                value "8";
                description "atm-nto1-vcc:";
            }
            enum atm-nto1-vpc {
                value "9";
                description "atm-nto1-vpc:";
```

```
            }
            enum ip-layer2 {
                value "10";
                description "ip-layer2:";
            }
            enum atm-1to1-vcc {
                value "11";
                description "atm-1to1-vcc:";
            }
            enum atm-1to1-vpc {
                value "12";
                description "atm-1to1-vpc:";
            }
            enum atm-aal5-pdu {
                value "13";
                description "atm-aal5-pdu:";
            }
            enum fr-port-mode {
                value "14";
                description "fr-port-mode:";
            }
            enum cesop {
                value "15";
                description "cesop:";
            }
            enum satop-e1 {
                value "16";
                description "satop-e1:";
            }
            enum satop-t1 {
                value "17";
                description "satop-t1:";
            }
            enum satop-e3 {
                value "18";
                description "satop-e3:";
            }
            enum satop-t3 {
                value "19";
                description "satop-t3:";
            }
            enum cesopsn-basic {
                value "20";
                description "cesopsn-basic:";
            }
            enum tdmoip_aal1 {
                value "21";
                description "tdmoip_aal1:";
```

```
            }
            enum cesopsn_tdm {
                value "22";
                description "cesopsn_tdm:";
            }
            enum tdmoip_aal2 {
                value "23";
                description "tdmoip_aal2:";
            }
            enum fr_dlci {
                value "24";
                description "fr_dlci:";
            }
            enum ip-interworking {
                value "25";
                description "ip-interworking:";
            }
            enum unsupport {
                value "26";
                description "unsupport:";
            }
        }
    }

    typedef pw-role {
        description "pw role.";
        type enumeration {
            enum primary {
                value "0";
                description "primary:";
            }
            enum backup {
                value "1";
                description "backup:";
            }
            enum bypass {
                value "2";
                description "bypass:";
            }
            enum multiHopOneSidePrimary {
                value "3";
                description "multiHopOneSidePrimary:";
            }
            enum multiHopOtherSidePrimary {
                value "4";
                description "multiHopOtherSidePrimary:";
            }
            enum multiHopOtherSideBackup {
```

```
                value "5";
                description "multiHopOtherSideBackup:";
            }
        }
    }

    typedef tunnelType {
        description "Indicates the type of tunnel used by the PW.";
        type enumeration {
            enum invalid {
                value "0";
                description "invalid tunnel type";
            }
            enum ldp {
                value "1";
                description "LDP LSP";
            }
            enum bgp {
                value "2";
                description "BGP LSP";
            }
            enum te {
                value "3";
                description "TE tunnel";
            }
            enum static_lsp {
                value "4";
                description "static lsp";
            }
            enum gre {
                value "5";
                description "GRE tunnel";
            }
            enum uni {
                value "6";
                description "uni tunnel";
            }
            enum tnl_group {
                value "7";
                description "tnl-group";
            }
            enum sub_te {
                value "8";
                description "TE sub tunnel";
            }
            enum sub_group {
                value "9";
                description "sub tunnel group";
```

```
            }
            enum 6over4 {
                value "10";
                description "manual IPv6 tunnel carry IPv4 traffic";
            }
            enum 6to4 {
                value "11";
                description "automatic IPv6 tunnel carry IPv4 traffic";
            }
            enum bgp_local_ifnet {
                value "12";
                description "BGP created mpls localifnet tunnel";
            }
            enum ldp6 {
                value "13";
                description "IPv6 LDP LSP";
            }
        }
    }

    typedef ifState {
        description "Interface state.";
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum plugOut {
                value "2";
                description "plugOut:";
            }
            enum notifyDown {
                value "3";
                description "notifyDown:";
            }
            enum downNotify {
                value "4";
                description "downNotify:";
            }
        }
    }

    typedef pwState {
        description "Indicates the status of the PW.";
```

```
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum backup {
                value "2";
                description "backup:";
            }
        }
    }

    typedef instanceType {
        description "Instance type. ";

        type enumeration {
            enum vpwsLocalccc {
                value "0";
                description "vpwsLocalccc:";
            }
            enum vpwsRemoteccc {
                value "1";
                description "vpwsRemoteccc:";
            }
            enum vpwsSvc {
                value "2";
                description "vpwsSvc:";
            }
            enum vpwsLdp {
                value "3";
                description "vpwsLdp:";
            }
            enum vpwsSwitch {
                value "4";
                description "vpwsSwitch:";
            }
            enum vpls {
                value "5";
                description "vpls:";
            }
        }
    }

    /* Grouping */
```

```
    grouping tdmParameter {
        description
            "Configure TDM parameter.";
        leaf tdmEncapsulateNumber {
            description "Number of encapsulated TDM frames.";
            config "true";
            type uint8 {
                range "1..40";
            }
        }
        leaf jitterBuffer {
            description "Depth of the TDM jitter buffer.";
            config "true";
            type uint16 {
                range "1000..64000";
            }
        }
        leaf rtpHeader {
            description
                "Whether or not the RTP header is added into the
                 transparently transported TDM frame.";
            config "true";
            type boolean;
        }
        leaf idleCode {
            description
                "Specifies the value of the idle code that is filled
                 manually when the jitter buffer underflow occurs.";
            config "true";
            type string {
                length "1..2";
                pattern "^((([1-9]|[a-f]|[A-F])([0-9]|[a-f]|[A-F])?)|
                         (0([0-9]|[a-f]|[A-F])?))$";
            }
        }
        leaf tdmSequenceNumber {
            description "Enable the seq-number option.";
            config "true";
            type boolean;
        }
        leaf payloadCompression {
            description
                "Specifies the dynamic bandwidth allocation for payload
                 compression.";
            config "true";
            type boolean;
        }
        leaf timeSlot {
```

```
            description
                "Specifies the time slot of the serial interface.";
            config "true";
            type uint8 {
                range "1..32";
            }
        }
    }

    grouping atmParameter {

        description "Configure ATM parameter.";

        leaf maxAtmCells {
            description "Maximum number of transmitted ATM cells.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf atmPackOvertime {
            description "Delay in packing ATM cells.";
            config "true";
            type uint16 {
                range "100..10000";
            }
        }
        leaf atmTransmitCell {
            description "ATM transmit cell.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf sequenceNumber {
            description
                "Enable the seq-number option.";
            config "true";
            type boolean;
        }
    }

    grouping speInfos {

        leaf speCount {
            description "Number of Spe.";
            config "false";
            type uint32;
```

```
        }

        list speInfo {

            config "false";
            key "spePwId spePeerIp";

            leaf spePwId {
                description
                    "Indicates the identifier of the PW.";
                type uint32;
            }
            leaf spePeerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                type inet:ip-address;
            }
        }
    }

    grouping vpwsPws {

        list vpwsPw {

            key "pwRole pwId";
            description "L2vpn vpws pw class.";

            leaf pwRole {
                description
                    "VPWS pw role:primary, backup,bypass.";
                config "true";
                type pw-role;

            }
            leaf pwId {
                description
                    "Indicates the identifier of the PW.";
                config "true";
                type uint32 {
                    range "1..4294967295";
                }
            }
            leaf peerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                config "true";
                type inet:ip-address;
            }
```

```
            leaf transmitLabel {
                description
                    "Indicates the label value of sent packets.";
                config "true";
                type uint32 {
                    range "0..1048575";
                }
            }
            leaf receiveLabel {
                description
                    "Indicates the label value of received packets.";
                config "true";
                type uint32 {
                    range "16..32767";
                }
            }
            leaf ctrlWord {
                description
                    "Enables the control word function. The control word
                    function is usually enabled on PWs with encapsulation
                    types being TDM, ATM or FR.

                    By default:
                    The control word function is enabled for TDM-, ATM-, or
                    Frame Relay-encapsulated PWs if PW profiles are not used.
                    If a PW profile is used, the control word function can
                    be enabled only after the control word is explicitly
                    specified. The control word function can be enabled for
                    PWs that use other types of encapsulation only after
                    the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
            }
            leaf vccvAbility {
                description
```

```
                    "Configures VC connectivity detection. VC connectivity
                    detection supports two modes: control word mode and
                    label alert mode.";

                config "true";
                default "true";
                type boolean;
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the L2VC. If no name
                    is specified for a tunnel policy, the default tunnel
                    policy is adopted. The LSP tunnel is preferred and only
                    one LSP is used for load balancing in the default tunnel
                    policy. If the name of the tunnel policy is specified but
                    no tunnel policy is configured, the default tunnel policy
                    is still adopted.";

                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf pwTemplateName {
                description
                    "Specifies the name of a PW template. You can set
                     attributes for a PW template, including the remote
                     peer, tunnel policy, and control word. When configuring
                     an LDP PW, you can directly apply the PW template rather
                     than specifying attributes for the PW.";

                config "true";
                type string {
                    length "1..19";
                }
            }
            leaf requestVlanId {
                description
                    "Indicates the requested VLAN ID.";
                config "true";
                type uint16 {
                    range "1..4094";
                }
            }
            leaf vlanTpId {
                description "Indicates the TPID of requested VLAN ID.";
                config "true";
                type string {
```

```
                    length "1..4";
                    pattern "^([0-9]|[a-f]|[A-F]){0,4}$";
                }
            }
            leaf pwTtl {
                description "Specify the TTL for PW.";
                config "true";
                type uint8 {
                    range "1..255";
                }
            }
            container tdmParameters {
                uses tdmParameter;
            }

            container atmParameters {
                uses atmParameter;
            }

            container vpwsLdpPwInfo {

                leaf interfaceName {
                    description
                        "Indicates the type and number of the AC
                         interface.";
                    config "false";
                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf ifState {
                    description "Indicates  status of the AC interface.";
                    config "false";
                    type ifState;
                }
                leaf sessionState {
                    description
                        "Indicates the status of the LDP session established
                         between both ends of the VC.";
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum down {
                            value "1";
                            description "down:";
```

```
                    }
                    enum up {
                        value "2";
                        description "up:";
                    }
                }
            }
            leaf integrativeAcState {
                description
                    "The integrative status of the AC.";
                config "false";
                type ifState;
            }
            leaf acState {
                description
                    "Indicates the status of the AC.";
                config "false";
                type ifState;
            }
            leaf pwState {
                description
                    "Indicates the status of the PW.";
                config "false";
                type pwState;
            }
            leaf pwId {
                description
                    "Indicates the identifier of the PW.";
                config "false";
                type uint32;
            }
            leaf encapType {
                description
                    "Indicates the encapsulation type of the PW.";
                config "false";
                type pw-encapsulation ;

            }
            leaf destination {
                description
                    "Indicates the LSR ID of the VC peer device.";
                config "false";
                type inet:ip-address;
            }
            leaf localGroupId {
                description "Indicates the local group ID.";
                config "false";
                type uint32;
```

```
                }
                leaf remoteGroupId {
                    description "Indicates the remote group ID.";
                    config "false";
                    type uint32;
                }
                leaf localVcLabel {
                    description "Indicates the local VC label.";
                    config "false";
                    type uint32;
                }
                leaf remoteVcLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32;
                }
                container tdmInfo {

                    description "TDM info";
                    config "false";

                    leaf localTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf remoteTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf jitterBuffer {
                        description "Depth of the TDM jitter buffer.";
                        config "false";
                        type uint8;
                    }
                    leaf idleCode {
                        description
                             "Indicates the idle code that is filled manually
                              when the jitter buffer underflow occurs.";
                        config "false";
                        type string {
                            length "0..3";
                        }
                    }
                    leaf localRtpHeaderEnable {
                        description
                             "Whether or not the RTP header is added into
```

```
                                  the transparently transported TDM frame.";
                            config "false";
                            type boolean;
                        }
                        leaf remoteRtpHeaderEnable {
                            description
                                 "Whether or not the RTP header is added into the
                                  transparently transported TDM frame.";
                            config "false";
                            type boolean;
                        }
                        leaf localBitRate {
                            description "Indicates the bit-rate of the local VC.";
                            config "false";
                            type uint16;
                        }
                        leaf remoteBitRate {
                            description "Indicates the bit-rate of the remoteVC.";
                            config "false";
                            type uint16;
                        }
                    }
                    container atmInfo {

                        description "TDM info";
                        config "false";

                        leaf maxAtmCells {
                            description "Maximum number of transmitted ATM cells.";
                            config "false";
                            type uint8 {
                                range "1..28";
                            }
                        }
                        leaf remoteMaxAtmCells {
                            description "Maximum number of transmitted ATM cells.";
                            config "false";
                            type uint8 {
                                range "0..28";
                            }
                        }
                        leaf atmPackOvertime {
                            description "Delay in packing ATM cells.";
                            config "false";
                            type uint16 {
                                range "100..10000";
                            }
                        }
```

```
                    leaf atmTransmitCell {
                        description "ATM transmit cell.";
                        config "false";
                        type uint16 {
                            range "1..28";
                        }
                    }
                    leaf sequenceNumber {
                        description
                             "Enable the seq-number option.By default, the
                              seq-number option is disabled.";
                        config "false";
                        type boolean;
                    }
                }
                leaf localFwdState {
                    description
                        "Indicates the status of the local forwarding table.";
                    config "false";
                    type enumeration {
                        enum notForwarding {
                            value "0";
                            description "notForwarding:";
                        }
                        enum forwarding {
                            value "1";
                            description "forwarding:";
                        }
                    }
                }
                leaf localStateCode {
                    description
                        "Indicates the status code of the local PW:
                        0x0: indicates that the local PW functioning as the
                             master PW is in the Up state.
                        0x20: indicates that the local PW functioning as the
                              backup PW is in the Up state.
                        0x1: indicates that the PW functioning as the master
                             PW and is in the Down state.
                        0x21: indicates that the PW functioning as the backup
                             PW and is in the Down state.";
                    config "false";
                    type uint32;
                }
                leaf remoteFwdState {
                    description
                        "Indicates the status of the remote forwarding
                         table.";
```

```
                    config "false";
                    type enumeration {
                        enum notForwarding {
                            value "0";
                            description "notForwarding:";
                        }
                        enum forwarding {
                            value "1";
                            description "forwarding:";
                        }
                    }
                }
                leaf remoteStateCode {
                    description
                        " Indicates the status code of the remote PW:
                        0x0: indicates that the remote PW functioning as the
                             master PW is in the Up state.
                        0x20: indicates that the remote PW functioning as the
                              backup PW is in the Up state.
                        0x1: indicates that the PW functioning as the master
                             PW and is in the Down state.
                        0x21: indicates that the PW functioning as the backup
                              PW and is in the Down state.";
                    config "false";
                    type uint32;
                }
                leaf isActive {
                    description
                        "Indicates whether or not the PW is in active state
                        (if so, user packets can be forwarded).";
                    config "false";
                    type boolean;
                }
                leaf isForwardExist {
                    description
                        "Indicates whether or not forwarding entries exist.";
                    config "false";
                    type boolean;
                }
                leaf linkState {
                    description
                        "Indicates the link status of the AC interface:
                        Up: indicates that the physical layer status of
                            the interface is functional.
                        Down: indicates that the physical layer of the
                              interface fails.";
                    config "false";
                    type enumeration {
```

```
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum down {
                            value "1";
                            description "down:";
                        }
                        enum up {
                            value "2";
                            description "up:";
                        }
                    }
                }
                leaf localVcMtu {
                    description "Indicates the MTU of the local VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteVcMtu {
                    description "Indicates the MTU of the remote VC.";
                    config "false";
                    type uint16;
                }
                leaf localVCCV {
                    description
                        "Indicates the type of VCCV that is supported
                         locally. By default, the control word function
                         is not enabled, and the supported VCCV type is
                         alert lsp-ping bfd, indicating that LSP ping
                         and BFD are supported for the alert channel.
                         If the control word function is enabled, the
                         VCCV type is cw alert lsp-ping bfd, indicating
                         that LSP ping and BFD are supported both for the
                         control word channel and the alert channel.";
                    config "false";
                    type string {
                        length "0..40";
                    }
                }
                leaf remoteVCCV {
                    description
                        "Indicates the type of VCCV that is supported
                         remotely. By default, the control word function
                         is not enabled, and the supported VCCV type is
                         alert lsp-ping bfd, indicating that LSP ping and
                         BFD are supported for the alert channel. If the
                         control word function is enabled, the VCCV type
```

```
                     is cw alert lsp-ping bfd, indicating that LSP ping
                     and BFD are supported both for the control word
                     channel and the alert channel.";

                config "false";
                type string {
                    length "0..40";
                }
            }
            leaf localCtrlWord {
                description
                    "Indicates whether or not the control word is enabled
                    on the local end.";
                config "false";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
            }
            leaf remoteCtrlWord {
                description
                    "Indicates whether or not the control word is enabled
                     on the remote end.";
                config "false";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
```

```
                }
                leaf tnlPolicyName {
                    description "Indicates the name of the tunnel policy.";
                    config "false";
                    type string {
                        length "1..39";
                    }
                }
                leaf pwTemplateName {
                    description "Indicates the name of the PW template.";
                    config "false";
                    type string {
                        length "1..19";
                    }
                }
                leaf priOrSec {
                    description
                        "Indicates whether the local status of the VC is
                         primary or secondary.";
                    config "false";
                    type enumeration {
                        enum primary {
                            value "0";
                            description "primary:";
                        }
                        enum secondary {
                            value "1";
                            description "secondary:";
                        }
                        enum bypass {
                            value "2";
                            description "bypass:";
                        }
                    }
                }
                leaf tunnelCount {
                    description
                        "Indicates that the PW uses one tunnel or token";
                    config "false";
                    type uint8;
                }
                container tunnelInfos {

                    config "false";
                    list tunnelInfo {

                        key "tunnelKey";
```

```
                        leaf tunnelKey {
                            description
                                "Indicates the ID of the tunnel used by the
                                 PW.";

                            type string {
                                length "0..21";
                            }
                        }
                        leaf tunnelType {
                            description
                                "Indicates the type of tunnel used by the
                                 PW.";
                            config "false";
                            type tunnelType;
                        }
                        leaf tunnelName {
                            description
                                "Indicates the name of the tunnel used by the
                                 PW.";
                            config "false";
                            type string {
                                length "0..64";
                            }
                        }
                        leaf publicNextHop {
                            description
                                "Indicates public next hop of a tunnel.";
                            config "false";
                            type inet:ip-address;
                        }
                    }

                }

                container speInfos {
                    config "false";
                    uses speInfos;
                }

                leaf createTime {
                    description
                        "Indicates how long the VC has been created for.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
```

```
            leaf upTime {
                description
                    "Indicates how long the VC keeps the Up state.
                     If the PW is currently in the Down state, the
                     value is 0.";
                config "false";
                type string {
                    length "1..80";
                }
            }
            leaf lastChgTime {
                description
                    "Indicates how long the VC status has remained
                     unchanged.";
                config "false";
                type string {
                    length "1..80";
                }
            }
            leaf pwLastUpTime {
                description
                    "Indicates the last time the VC was Up.";
                config "false";
                type yang:date-and-time;
            }
            leaf pwTotalUpTime {
                description
                    "Indicates the total duration the VC is Up.";
                config "false";
                type string {
                    length "1..80";
                }
            }
            leaf supportNotification {
                description
                    "Indicates whether or not the Notification
                     message is supported.";
                config "false";
                type boolean;
            }
        }

    }

}

grouping vplsPwInfo {
```

```
        leaf peerIp {
            description
                "Indicates the LSR ID of the VC peer device.";
            config "false";
            type inet:ip-address;
        }
        leaf pwId {
            description
                "Indicates the identifier of the PW.";
            config "false";
            type uint32;
        }
        leaf pwType {
            description "Type of the PW.label, QinQ,MEHVPLS";
            config "false";
            type enumeration {
                enum label {
                    value "0";
                    description "label:";
                }
                enum QinQ {
                    value "1";
                    description "QinQ:";
                }
                enum MEHVPLS {
                    value "2";
                    description "MEHVPLS:";
                }
            }
        }
        leaf sessionState {
            description
                "Indicates the status of the LDP session established
                 between both ends of the VC.";
            config "false";
            type enumeration {
                enum default {
                    value "0";
                    description "default:";
                }
                enum down {
                    value "1";
                    description "down:";
                }
                enum up {
                    value "2";
                    description "up:";
                }
```

```
            }
        }
        leaf pwState {
            description "Indicates the status of the PW.";
            config "false";
            type enumeration {
                enum down {
                    value "0";
                    description "down:";
                }
                enum up {
                    value "1";
                    description "up:";
                }
                enum backup {
                    value "2";
                    description "backup:";
                }
            }
        }
        leaf localVcLabel {
            description "Indicates the local VC label.";
            config "false";
            type uint32;
        }
        leaf remoteVcLabel {
            description "Indicates the remote VC label.";
            config "false";
            type uint32;
        }
        leaf tnlPolicyName {
            description
                "Indicates the name of the tunnel policy.";
            config "false";
            type string {
                length "1..39";
            }
        }
        leaf pwLastUpTime {
            description
                "Indicates the last time the VC was Up.";
            config "false";
            type yang:date-and-time;
        }
        leaf pwTotalUpTime {
            description
                "Indicates the total duration the VC is Up.";
            config "false";
```

```
            type string {
                length "1..80";
            }
        }
        container tunnelInfos {

            config "false";

            list tunnelInfo {

                key "tunnelKey";

                leaf tunnelKey {
                    description
                        "Indicates the ID of the tunnel used by the PW.";
                    type string {
                        length "0..21";
                    }
                }
                leaf tunnelType {
                    description
                        "Indicates the type of tunnel used by the PW.";
                    config "false";
                    type tunnelType;
                }
                leaf outIntf {
                    description
                        "Outbound interface.";
                    config "false";
                    type string {
                        length "0..256";
                    }
                }
                leaf tunnelName {
                    description
                        "Indicates the name of the tunnel used by the PW.";
                    config "false";
                    type string {
                        length "0..64";
                    }
                }
                leaf publicNextHop {
                    description "Assign public next hop of a tunnel.";
                    config "false";
                    type inet:ip-address;
                }
            }
```

```
        }
        container speInfos {
            config "false";
            uses speInfos;
        }
        leaf remoteGroupId {
            description "ID of the remote group.";
            config "false";
            type uint32;
        }
        leaf remoteMtu {
            description "Indicates the MTU of a remote VC.";
            config "false";
            type uint16;
        }
        leaf remoteVCCVcode {
            description "Indicates the VCCV of a remote VC.";
            config "false";
            type string {
                length "0..40";
            }
        }
        leaf remoteStateCode {
            description
                "Indicates the status of a remote VC, which can be:
                FORWARD: The remote VC is in the forwarding state.
                STANDBY: The remote VC is in the standby state.
                AC FAULT: The remote AC interface is faulty.
                PSN FAULT: The remote VC is faulty.
                NO FORWRD: The remote VC interface cannot forward packets
                           owing to other reasons. ";
            config "false";
            type enumeration {
                enum forward {
                    value "0";
                    description "forward:";
                }
                enum not-forward {
                    value "1";
                    description "not forward:";
                }
                enum standby {
                    value "2";
                    description "standby:";
                }
                enum ac-fault {
                    value "3";
                    description "ac fault:";
```

```
                }
                enum psn-fault {
                    value "4";
                    description "psn fault:";
                }
            }
        }
    }

    /* container */
    container l2vpncommon {

        container l2vpnGlobal {

            description "L2VPN golbal attribute.";

            leaf l2vpnEnable {
                description
                    "L2vpn enable flag.";
                type boolean;
                config "true";
                mandatory "true";
            }
            leaf vplsLoopDetectEnable {
                description
                    "Vpls mac withdraw loop detect enable flag.";
                type boolean;
                config "true";
            }
        }

        container pwTemplates {

            list pwTemplate {

                key "pwTemplateName";
                description "L2VPN pw template class.";

                leaf pwTemplateName {
                    description
                        "Specifies the PW template name. The value is a
                         case-sensitive string of 1 to 19 characters without
                         blank space.";
                    config "true";
                    type string {
                        length "1..19";
                    }
                }
```

```
                leaf peerAddr {
                    description
                        "Assign a peer IP address to a PW template.";
                    config "true";
                    type inet:ip-address;
                }
                leaf mtu {
                    description
                        "Configure the mtu value for PW template, 46 to 9600.";
                    config "true";
                    default "1500";
                    type uint16 {
                        range "46..9600";
                    }
                }
                leaf ctrlWord {
                    description
                        "Enable the control word in a PW template.";
                    config "true";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf tunnelPolicy {
                    description
                        "Configure a tunnel policy for a PW template.";
                    config "true";
                    type string {
                        length "1..39";
                    }
                }
                uses tdmParameter;

                uses atmParameter;

            }
        }
```

```
        container notMatchRemoteLdpInfos {

            config "false";
            list notMatchRemoteLdpInfo {

                key "pwId peerIp encapsulateType";

                leaf pwId {
                    description
                        "After an ID is set for a VC, it cannot be changed.
                         Different VCs have different IDs.";
                    type uint32;
                }
                leaf peerIp {
                    description "Indicates the peer ip of the VC peer device.";
                    type inet:ip-address;
                }
                leaf encapsulateType{
                    description "Indicates the encapsualtion VC peer device.";
                    type pw-encapsulation;
                }
                leaf remoteLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32 ;
                }
                leaf remoteGroupId {
                    description "ID of the remote group.";
                    config "false";
                    type uint32;
                }
                leaf remoteMtu {
                    description "Indicates the MTU of a remote VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteStateCode {
                    description
                        "Indicates the status of a remote VC, which can be:
                            FORWARD: The remote VC is in the forwarding state.
                            STANDBY: The remote VC is in the standby state.
                            AC FAULT: The remote AC interface is faulty.
                            PSN FAULT: The remote VC is faulty.
                            NO FORWRD: The remote VC interface cannot forward
                                       packets owing to other reasons.";
                    config "false";
                    type enumeration {
                        enum forward {
```

```
                        value "0";
                        description "forward:";
                    }
                    enum not-forward {
                        value "1";
                        description "not forward:";
                    }
                    enum standby {
                        value "2";
                        description "standby:";
                    }
                    enum ac-fault {
                        value "3";
                        description "ac fault:";
                    }
                    enum psn-fault {
                        value "4";
                        description "psn fault:";
                    }
                }
            }
        }

    }
}

container l2vpnvpws {

    container vpwsStatisticInfo {

        container vpwsLdpAcStatInfo {

            leaf totalLdpAcNum {
                description "Total number of L2VPN VPWS LDP AC.";
                config "false";
                type uint32;
            }
            leaf upLdpAcNum {
                description "Up number of L2VPN VPWS LDP AC.";
                config "false";
                type uint32;
            }
            leaf downLdpAcNum {
                description "Down number of L2VPN VPWS LDP AC.";
                config "false";
                type uint32;
            }
        }
```

```
            container vpwsLdpPwStatInfo {

                leaf totalLdpPwNum {
                    description
                        "Indicates the total number of established LDP PWs";
                    config "false";
                    type uint32;
                }
                leaf upLdpPwNum {
                    description "Number of LDP PWs in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downLdpPwNum {
                    description "Number of LDP PWs in the down state.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsLdpPwRemoteStatInfo {

                leaf remoteVcNum {
                    description
                        "Indicates the total number of created remote LDP
                         PWs.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsSwitchInstanceStatInfo {

                leaf totalSwitchInstanceNum {
                    description
                        "Indicates the total number of established switch-vc";
                    config "false";
                    type uint32;
                }
                leaf upSwitchInstanceNum {
                    description "Number of switch-vc in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downSwitchInstanceNum {
                    description "Number of switch-vc in the down state.";
                    config "false";
                    type uint32;
```

```
                }
            }

        }

        container vpwsInstances {

            list vpwsInstance {

                key "instanceName instanceType";

                description "L2vpn vpws instance class.";

                leaf instanceName {

                    description "Specifies VPWS instance name.";
                    config "true";

                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf instanceType {
                    description "VPWS instance type:ldp,localccc.";
                    config "true";
                    type instanceType;
                }
                leaf encapsulateType {
                    type pw-encapsulation;
                }
                leaf description {
                    description "Specifies a description for the VC.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf instanceState {
                    description "VPWS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
```

```
                        }
                        enum adminDown {
                            value "2";
                            description "adminDown:";
                        }
                    }
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the instance keeps the Up state.
                         If the PW is currently in the Down state, the value
                         is 0.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf totalUpTime {
                    description
                        "Indicates the total duration the instance is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }

                container tdmParameters {
                    uses tdmParameter;
                }

                container atmParameters {
                    uses atmParameter;
                }

                container l2vpnAcs {

                    list l2vpnAc {

                        key "interfaceName";
                        description "L2VPN ac class.";

                        leaf interfaceName {
                            description "Specifies the AC interface name.";
                            config "true";
                            type leafref {
                                path "/if:interfaces/if:interface/if:name";
                            }
                        }
                        leaf state {
                            description "Indicates the status of the AC.";
```

```
                config "false";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum down {
                        value "1";
                        description "down:";
                    }
                    enum up {
                        value "2";
                        description "up:";
                    }
                }
            }

            container l2vpnPipe {

                description "L2VPN pipe mode.";

                leaf pipeMode {
                    description "Pipe mode.";
                    default "uniform";
                    type enumeration {
                        enum pipe {
                            value "0";
                            description "pipe:";
                        }
                        enum shortPipe {
                            value "1";
                            description "shortPipe:";
                        }
                        enum uniform {
                            value "2";
                            description "uniform:";
                        }
                    }
                }
            }

            container ifLinkProtocolTran {

                description "lacp status";

                leaf protocolLacp {
                    description "lacp status";
                    config "true";
```

```
                                type enumeration {
                                    enum enable {
                                        value "0";
                                        description "enable:";
                                    }
                                    enum disable {
                                        value "1";
                                        description "disable:";
                                    }
                                }
                            }
                            leaf protocolLldp {
                                description "lldp status";
                                config "true";
                                type enumeration {
                                    enum enable {
                                        value "0";
                                        description "enable:";
                                    }
                                    enum disable {
                                        value "1";
                                        description "disable:";
                                    }
                                }
                            }
                            leaf protocolBpdu {
                                description "bpdu status";
                                config "true";
                                type enumeration {
                                    enum enable {
                                        value "0";
                                        description "enable:";
                                    }
                                    enum disable {
                                        value "1";
                                        description "disable:";
                                    }
                                }
                            }
                            leaf protocolCdp {
                                description "cdp status";
                                config "true";
                                type enumeration {
                                    enum enable {
                                        value "0";
                                        description "enable:";
                                    }
                                    enum disable {
```

```
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                    leaf protocolUdld {
                        description "udld status";
                        config "true";
                        type enumeration {
                            enum enable {
                                value "0";
                                description "enable:";
                            }
                            enum disable {
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                }

            }

        }

        container vpwsPws {
            uses vpwsPws;
        }

        container reliabilitys {

            list reliability {

                key "pwRedundancyMode";

                leaf pwRedundancyMode {
                    description
                        "Specifies the redundancy mode of VPWS
                         instance.";

                    config "true";
                    type enumeration {
                        enum frr {
                            value "0";
                            description "frr:";
                        }
                        enum masterSlave {
                            value "1";
```

```
                        description "masterSlave:";
                    }
                    enum independent {
                        value "2";
                        description "independent:";
                    }
                }
            }
            leaf switchover {
                description
                    "Specifies switches traffic from the primary
                     PW to the secondary PW.";
                config "true";
                type boolean;
            }
            leaf dualReceive {
                description
                    "Specifies enables a UPE interface to receive
                     packets from both the primary and secondary
                     PWs.";
                config "true";
                type boolean;
            }
            container reRoute {

                description "L2vpn vpws pw reroute class.";

                leaf reRoutePolicy {
                    description "Specifies the Policy of Reroute.";
                    config "true";
                    type enumeration {
                        enum delay {
                            value "0";
                            description "delay:";
                        }
                        enum immediately {
                            value "1";
                            description "immediately:";
                        }
                        enum never {
                            value "2";
                            description "never:";
                        }
                    }
                }
                leaf delayTime {
                    description
                        "Specifies the delay for switching traffic
```

```
                         back to the primary PW. ";
                    config "true";
                    type uint16 {
                        range "10..600";
                    }
                }
                leaf resumeTime {
                    description
                        "Specifies the time after which the peer PE
                         on the secondary PW is notified that the
                         local PE is recovered from the fault. ";
                    config "true";
                    type uint16 {
                        range "0..600";
                    }
                }
                leaf lastReRouteReason {
                    description
                        "Specifies the reason of Last Reroute.";
                    config "false";
                    type string {
                        length "0..100";
                    }
                }
                leaf lastReRouteTime {
                    description
                        "Specifies the time of Last Reroute.";
                    config "false";
                    type string {
                        length "1..60";
                    }
                }
                leaf delayResidual {
                    description
                        "Specifies the residual delay time for
                         switching traffic back to the primary PW.
                        ";
                    config "false";
                    type uint32;
                }
                leaf resumeResidual {
                    description
                        "Specifies the residual time after which
                         the peer PE on the secondary PW is
                         notified that the local PE is recovered
                         from the fault. ";
                    config "false";
                    type uint32;
```

```
                        }
                    }

                }

            }

        }

    }

    container vpwsSwitchInstances {

        list vpwsSwitchInstance {

            key "instanceName instanceType";

            description "L2vpn vpws instance class.";

            leaf instanceName {
                description "Specifies VPWS instance name.";
                config "true";
                type string {
                    length "1..31";
                }
            }
            leaf instanceType {
                description "VPWS instance type:vpwsSwitch.";
                config "true";
                type instanceType;
            }
            leaf encapsulateType {
                description "VPWS instance encapsulation type.";
                config "true";
                default "ethernet";
                type pw-encapsulation;
            }
            leaf switchType {
                description
                    "VPWS switch instance type:ldp2ldp,ldp2svc.";
                config "true";
                default "ldp2ldp";
                type enumeration {
                    enum svc2svc {
                        value "0";
                        description "svc2svc:";
                    }
                    enum ldp2svc {
```

```
                        value "1";
                        description "ldp2svc:";
                    }
                    enum ldp2ldp {
                        value "2";
                        description "ldp2ldp:";
                    }
                    enum upe {
                        value "3";
                        description "upe:";
                    }
                }
            }
            leaf ctrlWordTrans {
                description
                    "Transparent transmission of control word If BFD
                     is enabled to monitor dynamic multi-hop PWs,
                     transparent transmission of control word needs
                     to be configured on the SPE. Otherwise, the BFD
                     negotiation fails. By default, transparent
                     transmission of control word is disabled.";
                config "true";
                type boolean;
                default "false";
            }
            leaf controlWord {
                description
                    "Enables the control word function. The control
                     word function is usually enabled on PWs with
                     encapsulation types being TDM, ATM or FR.
                     By default:
                     The control word function is enabled for TDM-,
                     ATM-, or Frame Relay-encapsulated PWs if PW
                     profiles are not used. If a PW profile
                     is used, the control word function can be
                     enabled only after the control word is explicitly
                     specified. The control word function can be enabled
                     for PWs that use other types of encapsulation only
                     after the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
```

```
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf instanceState {
                    description "VPWS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
                        }
                        enum adminDown {
                            value "2";
                            description "adminDown:";
                        }
                    }
                }
                leaf createTime {
                    description
                        "Indicates how long the VC has been created for.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf upTime {
                    description
                        "Indicates how long the VC keeps the Up state. If the
                         PW is currently in the Down state, the value is 0.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf lastChgTime {
                    description
                        "Indicates how long the VC status has remained
                         unchanged.";
                    config "false";
                    type string {
```

```
                            length "1..80";
                        }
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the instance keeps the Up state.
                         If the PW is currently in the Down state, the value
                         is 0.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf totalUpTime {
                    description
                        "Indicates the total duration the instance is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }

                container vpwsPws {
                    uses vpwsPws;
                }

            }

        }
    }

    container l2vpnvpls {

        container vplsStatisticInfo {

            container vplsInstStatisticsInfo {

                leaf totalVsiNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf vsiUpNum {
                    config "false";
                    type uint32;
                }
                leaf vsiDownNum {
                    config "false";
                    type uint32;
                }
```

```
                leaf ldpModeNum {
                    config "false";
                    type uint32;
                }
                leaf bgpVsiNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdVsiNum {
                    config "false";
                    type uint32;
                }
                leaf unspecifiedNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsPwStatisticsInfo {

                leaf totalPwNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf upPwNum {
                    config "false";
                    type uint32;
                }
                leaf downPwNum {
                    config "false";
                    type uint32;
                }
                leaf ldpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdPwNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsAcStatisticsInfo {
```

```
                leaf totalVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf upVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf downVplsAcNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsTnlRefInfos {

                config "false";
                list vplsTnlRefInfo {

                    key "tnlPolicyName";

                    leaf tnlPolicyName {
                        description "None";
                        config "false";
                        type string {
                            length "1..39";
                        }
                    }
                    container tnlVsiRefInfos {

                        list tnlVsiRefInfo {

                             key "instanceName";

                             leaf instanceName {
                                config "false";
                                type string {
                                    length "1..31";
                                }
                            }
                        }

                    }

                }

            }
```

```
            container vplsLoopDetectStaticInfo {

                 leaf totalVplsLoopDetectNum {
                    config "false";
                    type uint32;
                }
            }

        }

        container vplsInstances {

            list vplsInstance {

                key "instanceName";

                leaf instanceName {
                    description
                        "Specifies VPLS instance name.";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf description {
                    description
                        "Specify the VPLS instance description.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf memberDiscoveryMode {
                    description
                        "The VPLS member discovery mode for a created VSI.";
                    config "true";
                    default "default";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum auto {
                            value "1";
                            description "auto:";
                        }
                        enum static {
                            value "2";
```

```
                        description "static:";
                    }
                    enum bdmode {
                        value "3";
                        description "bd mode:";
                    }
                }
            }
            leaf encapsulateType {
                description "VPWS instance encapsulation type.";
                config "true";
                default "vlan";
                type pw-encapsulation;
            }
            leaf mtuValue {
                description
                    "MTU specified in dynamic PW signaling negotiation.";
                config "true";
                default "1500";
                type uint16 {
                    range "328..65535";
                }
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the VSI. If no
                     name is specified for a tunnel policy, the default
                     tunnel policy is adopted. The LSP tunnel is
                     preferred and only one LSP is used for load balancing
                     in the default tunnel policy. If the name of the
                     tunnel policy is specified but no tunnel policy is
                     configured, the default tunnel policy is still adopted.
                     ";
                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf shutdown {
                description
                    "Sometimes, because of service debugging or service
                     halt, a VSI can be disabled for function
                     modification.";
                config "true";
                default "false";
                type boolean;
            }
            leaf isolateSpoken {
```

```
                    description
                        "The isolate spoken command enables forwarding
                         isolation between AC interfaces, between UPE
                         PWs, and between ACs and UPE PWs on a VSI. The
                         undo isolate spoken command disables forwarding
                         isolation";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf unknownUnicastAction {
                    description
                        "Specifies the processing mode for received unknown
                         unicast frames.";
                    config "true";
                    type enumeration {
                        enum broadcast {
                            value "0";
                            description "broadcast:";
                        }
                        enum drop {
                            value "1";
                            description "drop:";
                        }
                        enum local-handle {
                            value "2";
                            description "local-handle:";
                        }
                        enum drop-learn {
                            value "3";
                            description "drop-learn:";
                        }
                    }
                }
                leaf macLearnEnable {
                    description
                        "Enables MAC address learning on a VSI.";
                    config "true";
                    default "true";
                    type boolean;
                }
                leaf macLearnStyle {
                    description
                        "Sets the MAC address learning style of a VSI.By
                         default, MAC address learning style is unqualify.
                         Currently, the VRP supports only the unqualified
                         mode.";
```

```
                    config "true";
                    default "unqualify";
                    type enumeration {
                        enum qualify {
                            value "0";
                            description "qualify:";
                        }
                        enum unqualify {
                            value "1";
                            description "unqualify:";
                        }
                    }
                }
                leaf macAgeTimer {
                    description
                        "Sets the aging time of MAC address entries in a VSI.
                         By default, the aging time of MAC address entries in
                         a VSI is the global aging time. You can set the global
                         aging time by the command mac-address aging-time
                         (system view).";
                    config "true";
                    type uint32 {
                        range "0..1000000";
                    }
                }
                leaf totalAcService {
                    description
                        "Total number of interface in the instance.";
                    config "false";
                    type uint32;
                }
                leaf createTime {
                    description
                        "Indicates how long the VSI has been created for.";
                    config "false";
                    type string {
                        length "1..60";
                    }
                }
                leaf vsiState {
                    description "VPLS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
```

```
                        value "1";
                        description "up:";
                    }
                    enum adminDown {
                        value "2";
                        description "adminDown:";
                    }
                }
            }
            leaf ignoreAcStateEffect {
                description
                    "After the ignore-ac-state command is configured,
                     the VSI status is not subject to changes in the
                     status of the AC. That is, a VSI can go Up even
                     if no AC is connected to the VSI.";

                config "false";
                default "false";
                type boolean;
            }

            container vsiPipe {

                leaf pipeMode {
                    description "Pipe mode";
                    config "true";
                    default "uniform";
                    type enumeration {
                        enum pipe {
                            value "0";
                            description "pipe:";
                        }
                        enum shortPipe {
                            value "1";
                            description "shortPipe:";
                        }
                        enum uniform {
                            value "2";
                            description "uniform:";
                        }
                    }
                }
                leaf serviceClass {
                    description "service class";
                    config "true";
                    default "be";
                    type enumeration {
                        enum be {
```

```
                            value "0";
                            description "be:";
                        }
                        enum af1 {
                            value "1";
                            description "af1:";
                        }
                        enum af2 {
                            value "2";
                            description "af2:";
                        }
                        enum af3 {
                            value "3";
                            description "af3:";
                        }
                        enum af4 {
                            value "4";
                            description "af4:";
                        }
                        enum ef {
                            value "5";
                            description "ef:";
                        }
                        enum cs6 {
                            value "6";
                            description "cs6:";
                        }
                        enum cs7 {
                            value "7";
                            description "cs7:";
                        }
                    }
                }
                leaf color {
                    description "service color";
                    config "true";
                    default "green";
                    type enumeration {
                        enum green {
                            value "0";
                            description "green:";
                        }
                        enum yellow {
                            value "1";
                            description "yellow:";
                        }
                        enum red {
                            value "2";
```

```
                            description "red:";
                        }
                    }
                }
                leaf dsName {
                    description "domain name";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
            }

            container vplsLdpInst {

                leaf vsiId {
                    description
                        "After an ID is set for a VSI, it cannot be
                         changed. Different VSIs have different IDs.";
                    config "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf macWithdraw {
                    description
                        "Configures a VSI to delete the local MAC
                         addresses and informs all the remote peers
                         of the deletion when an AC fault or a UPE
                         fault occurs and the VSI remains Up.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf ifChgWithdraw {
                    description
                        "Configures PEs to send LDP MAC Withdraw
                         messages to all peers when the status of
                         the AC interface bound to the VSI changes.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf upeUpeMacWithdraw {
                    description
                        "Configures an NPE to forward the LDP MAC
                         Withdraw messages received from a UPE to
                         other UPEs.";
```

```
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf upeNpeMacWithdraw {
                        description
                            "Configures an NPE to forward the LDP MAC
                             Withdraw messages received from UPEs to
                             other NPEs.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf npeUpeMacWithdraw {
                        description
                            "Configures an NPE to forward the LDP MAC
                             Withdraw messages received from other NPEs
                             to UPEs.";
                        config "true";
                        type boolean;
                    }
                    container vplsLdpPws {

                        list vplsLdpPw {

                            key "peerIp pwId pwEncapType";

                            leaf peerIp {
                                description "Specifies the LSR ID of the peer PE
.";

                                config "true";
                                type inet:ip-address;
                            }
                            leaf pwId {
                                description
                                    "Indicates the identifier of the PW. Default
,

                                     we may use vsiId as the pwId. Sometimes we
may

                                     create pw to different pw that the pwId not
                                     match our vsiId, so it must specify the pwI
d.";


                                config "true";
                                type uint32 {
                                    range "1..4294967295";
                                }
                            }
                            leaf pwEncapType {
                                description
                                    "Indicates the encapsulation of the PW.
                                     Default, we may use encapsulateType as
```

```
                         the pwEncapType. Sometimes we may create
                         pw that the pwEncapType not match our
                         encapsulateType, so it must specify the
                         pwEncapType.";
                    config "true";
                    type pw-encapsulation;
                }
                leaf pwRole {
                    description
                        "VPLS pw role:primary, secondary.";
                    config "true";
                    default "primary";
                    type pw-role;
                }
                leaf pwName {
                    description
                        "Specifies the name of a PW, which is used
                         to distinguish the PW from other PWs. The
                         PW name must be unique in the same VSI,
                         but can be the same as the PW names in
                         other VSIs. ";
                    config "true";
                    type string {
                        length "1..15";
                    }
                }
                leaf ifParaVCCV {
                    description
                        "Deletes the VCCV byte (an interface
                         parameter) in the Mapping packet.";
                    config "true";
                    default "true";
                    type boolean;
                }
                leaf isUpe {
                    description "set VPLS PW as upe.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf tnlPolicyName {
                    description
                        "Specifies a tunnel policy name for the
                         VPLS PW. If no name is specified for a
                         tunnel policy, the default tunnel policy
                         is adopted. The LSP tunnel is preferred
                         and only one LSP is used for load
                         balancing in the default tunnel policy.
```

```
                             If the name of the tunnel policy is
                             specified but no tunnel policy is
                             configured, the default tunnel policy is
                             still adopted.";
                        config "true";
                        type string {
                            length "1..39";
                        }
                    }

                    container vplsLdpPwInfo {
                        config "false";
                        uses vplsPwInfo;
                    }
                }

            }
        }

        container vplsBgpAdInst {

            leaf vplsId {
                description
                    "Specifies the vpls id. The value is a
                     case-sensitive string of 3 to 21 characters
                     without blank space.";
                config "true";
                type string {
                    length "1..21";
                }
            }

            leaf bgpAdRd {
                description
                    "Specifies the Route Distinguisher. The value is
                     a case-sensitive string of 3 to 21 characters
                     without blank space.";
                config "false";
                type string {
                    length "1..21";
                }
            }

            leaf vsiId {
                description
                    "Specifies the negotiation ip address of the
                     local PE.";
                config "false";
```

```
                        type inet:ip-address;
                    }

                    container vpnTargets {
                        description "BGP-AD vpn-targets.";
                        list vpnTarget {
                            key "vpnRTValue";
                            description "BGP AD vpn targets.";

                            leaf vpnRTValue {

                              description
                                "Vpn-target:
                                 adds VPN target extended community attribute
                                 to the export or import VPN target extended
                                 community list. The vpn-target can be
                                 expressed in either of the following formats:
                                 (1)16-bit AS number:32-bit user-defined number
                                    For example, 1:3. The AS number ranges from
                                    0 to 65535. The user-defined number ranges
                                    from 0 to 4294967295. The AS number and the
                                    user-defined number cannot be 0s at the same
                                    time. That is, a VPN target cannot be 0:0.
                                 (2)32-bit IP address:16-bit user-defined number
                                    For example, 192.168.122.15:1. The IP addres
s

                                    ranges from 0.0.0.0 to 255.255.255.255. The
                                    user-defined number ranges from 0 to 65535."
;

                              mandatory "true";
                              type string {
                                length "3..21";
                              }
                            }

                            leaf vrfRTType {
                              description
                                "Specifies the vpn target type,
                                 export-extcommunity: specifies the extended
                                 community attributes carried in routing
                                 information to be sent. import-extcommunity:
                                 receives routing information carrying
                                 specified extended community attributes.";
                               mandatory "true";
                               type enumeration {
                               enum export_extcommunity {
                                 value "0";
                                 description "export-extcommunity:";
                               }
```

```
                           enum import_extcommunity {
                             value "1";
                             description "import-extcommunity:";
                           }
                           enum both {
                             value "2";
                             description
                               "export-extcommunity &
                                import-extcommunity:";
                           }
                         }
                       }
                    }
                }

                container bgpAdPeerInfos {
                    config "false";

                    list bgpAdPeerInfo {

                        key "peerRouterID";
                        leaf peerRouterID {
                            description
                                "The Router ID of the remote router.";
                            type inet:ip-address;
                        }

                        leaf vplsId {
                            description
                                "The vpls id. The value is a case-sensitive
                                 string of 3 to 21 characters without blank
                                 space.";
                            type string {
                                length "1..21";
                            }
                        }

                        leaf sourceAII {
                            description
                                "The source AII of the remote PE.";
                            type inet:ip-address;
                        }

                        leaf targetAII {
                            description
                                "The target AII of the remote PE.";
                            type inet:ip-address;
                        }
```

```
                        leaf peerType {
                          description
                            "Specifies the peer type.Static,Dynamic.";

                          type enumeration {
                                enum static {
                                  value "0";
                                  description "Static pw";
                                }
                                enum dynamic {
                                  value "1";
                                  description "Dynamic pw";
                                }
                          }
                        }

                        container bgpAdPwInfo {
                            config "false";

                            uses vplsPwInfo;
                        }

                    }
                }

            }

            container vplsBgpInst {

                leaf bgpRd {
                    description
                        "Specifies the Route Distinguisher. The value is a
                        case-sensitive string of 3 to 21 characters without
                        blank space.";
                    type string {
                        length "1..21";
                    }
                }

                leaf ignoreMtu {
                    description
                        "Ignore the mtu when negotiate pw.";
                    type boolean;
                }

                container vpnTargets {
                    description "BGP vpn-targets";
                    list vpnTarget {
```

```
                            key "vpnRTValue";
                            description
                                "BGP vpn targets";

                            leaf vpnRTValue {

                              description
                                "Vpn-target: adds VPN target extended community
                                 attribute to the export or import VPN target
                                 extended community list. The vpn-target can be
                                 expressed in either of the following formats:
                                (1)16-bit AS number:32-bit user-defined number
                                   For example, 1:3. The AS number ranges from
                                   0 to 65535. The user-defined number ranges
                                   from 0 to 4294967295. The AS number and the
                                   user-defined number cannot be 0s at the same
                                   time. That is, a VPN target cannot be 0:0.
                                (2)32-bit IP address:16-bit user-defined number
                                   For example, 192.168.122.15:1. The IP address
                                   ranges from 0.0.0.0 to 255.255.255.255. The
                                   user-defined number ranges from 0 to 65535.";

                              mandatory "true";
                              type string {
                                length "3..21";
                              }
                            }

                            leaf vrfRTType {
                              description
                                "Specifies the vpn target type,
                                 export-extcommunity: specifies the extended
                                 community attributes carried in routing
                                 information to be sent.
                                 import-extcommunity: receives routing
                                 information carrying specified extended
                                 community attributes.";

                              mandatory "true";
                              type enumeration {
                                    enum export_extcommunity {
                                      value "0";
                                      description "export-extcommunity:";
                                    }
                                    enum import_extcommunity {
                                      value "1";
                                      description "import-extcommunity:";
                                    }
```

```
                                   enum both {
                                     value "2";
                                     description "export-extcommunity &
                                                  import-extcommunity:";
                                   }
                             }
                           }
                       }
                   }

                   container bgpSite {

                       leaf siteId {
                           description
                               "Specifies the ID of the site.";
                           mandatory "true";
                           type uint16 {
                               range "1..65535";
                           }
                       }
                       leaf siteRange {
                           description
                               "Specifies the ID of the site range.";
                           type uint16 {
                               range "1..65535";
                           }
                       }
                       leaf defaultOffset {
                           description
                               "Specifies the default offset of the site ID.";
                           type uint8 {
                               range "0..1";
                           }
                       }
                   }

                   container bgpPeerInfos {
                       config "false";

                       list bgpPeerInfo {

                           key "siteId";

                           leaf siteId {
                               description
                                   "The site id of the peer.";
                               type uint16 {
                                   range "1..65535";
```

```
                            }
                        }

                        container bgpPwInfo {
                            config "false";

                            uses vplsPwInfo;
                        }

                    }
                }

            }

            container vplsAcs {

                list vplsAc {

                    key "interfaceName";

                    leaf interfaceName {
                        description "Specifies the AC interface name. ";
                        config "true";
                        type leafref {
                            path "/if:interfaces/if:interface/if:name";
                        }
                    }
                    leaf hubModeEnable {
                        description
                            "Change the VSI attribute of the local
                             interface from spoke to hub.By default,
                             the AC side of a VSI has the attribute
                             of spoke, and the PW side of a VSI has
                             the attribute of hub.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf state {
                        description
                            "Indicates the status of the AC.";
                        config "false";
                        type ifState;
                    }
                    leaf lastUpTime {
                        description
                            "Indicates how long the AC keeps the Up state.
                             If the AC is currently in the Down state, the
```

```
                             value is 0.";
                        config "false";
                        type yang:date-and-time;
                    }
                    leaf totalUpTime {
                        description
                            "Indicates the total duration the AC is Up.";
                        config "false";
                        type string {
                            length "1..60";
                        }
                    }
                    container ifLinkProtocolTran {

                        description "lacp status";

                        leaf protocolLacp {
                            description "lacp status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolLldp {
                            description "lldp status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolBpdu {
                            description "bpdu status";
                            config "true";
                            type enumeration {
```

```
                        enum enable {
                            value "0";
                            description "enable:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                    }
                }
                leaf protocolCdp {
                    description "cdp status";
                    config "true";
                    type enumeration {
                        enum enable {
                            value "0";
                            description "enable:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                    }
                }
                leaf protocolUdld {
                    description "udld status";
                    config "true";
                    type enumeration {
                        enum enable {
                            value "0";
                            description "enable:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                    }
                }
            }
        }

    }

    container vplsLoopDetectInfo {

        description "L2vpn admin vsi binding class.";

        leaf lastLoopType {
```

```
                        description
                            "Indicates the last mac withdraw loop type.";
                        config "false";
                        type enumeration {
                            enum detect-loop {
                                value "0";
                                description "detect loop:";
                            }
                            enum exceed-max-hop {
                                value "1";
                                description "exceed max hop:";
                            }
                        }
                    }
                    leaf sendPeer {
                        description
                            "Indicates the mac withdraw send peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf receivePeer {
                        description
                            "Indicates the mac withdrawreceive peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf lastLoopTime {
                        description
                            "Indicates the last mac withdraw loop time.";
                        config "false";
                        type string {
                            length "1..80";
                        }
                    }
                }

            }

        }

    }
}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

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

8. References

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

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

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

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Haibo Wang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: rainsword.wang@huawei.com

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

Email: lizhenbin@huawei.com