Network Working Group   J. Medved
Internet-Draft   A. McLachlan
Intended status: Informational   D. Meyer
Expires: January 4, 2013   Cisco Systems
                                                           July 03, 2012

MPLS-TP PseudoWire Configuration using OpenFlow 1.3
draft-medved-pwe3-of-config-00

Abstract

This document describes a method by which MPLS-TP Pseudowires (PW) can be configured using OpenFlow 1.3. In addition to the provisioning of the PWs this document also specifies how to enact OAM for these PWs using standard IETF conventions defined by the GAL label method. The primary goal of this document is to show a simple and yet flexible method using tools from the emerging SDN toolkit as opposed to a vendor driven provisioning system.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

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].

1.2. Terminology

This document uses the following terminology:

Term	Definition
AC:	Attachment Circuit
ACH:	Associated Channel Header
BFD:	Bidirectional Forwarding Detection
CE:	Customer Edge
G-ACh:	Generic Associated Channel
GAL:	G-ACh Label
LER:	Label Edge Router
LSP:	Label Switch Path
LSR:	Label Switch Router
MPLS:	Multiprotocol Label Switching
MPLS-TP:	MPLS Transport Profile
MPLS-TP P:	MPLS-TP Provider LSR
MPLS-TP PE:	MPLS-TP Provider Edge LSR
PDU:	Protocol Data Unit
PG:	Port Group
PSN:	Packet Switching Network
PW:	Pseudowire
OAM:	Operations, Administration, and Maintenance
OAM Engine:	Operations, Administration, and Maintenance Engine
OF:	OpenFlow
SDN:	Software Defined Networks
VP:	Virtual Port

1.3. Overview

MPLS-TP provides a relatively light weight layer 2 transport technology by leveraging elements of existing transport platforms and a subset of the more recent MPLS protocol standards. PWs are configured as bi-directional paths over the MPLS-TP network, usually by an external management platform. At present no open standards exist to provision these PWs, and therefore there is a reliance on vendor specific provisioning platforms.

This document describes a mechanism that uses the emerging OpenFlow standard ([OF-1.3.0]) to provision PWs and PW OAM in a TP environment. Naturally the implementation of OpenFlow will be required on the TP switch, as would an OAM Engine, the functions of which are described in this document. In addition, an OpenFlow Controller will be required for the provisioning functions.

Because OpenFlow is proposed as an open standard, it enables Service Providers to adopt a more consolidated approach to provisioning. An OpenFlow Controller can be common to a number of different elements in the network, as being driven by current industry SDN (Software Defined Networks) developments.

This document uses the reference MPLS-TP architecture defined in [RFC5921], which is shown in the following figure:

```
            |<----------------- Client Layer ------------------->|
            |                                                    |
            |          |<-------- Pseudowire -------->|          |
            |          |      encapsulated, packet    |          |
            |          |      transport service       |          |
            |          |                              |          |
            |          |          Transport           |          |
            |          |    |<------ LSP ------->|    |          |
            |          V    V                    V    V          |
            V  ^       +----+      +------+      +----+       ^  V
      +-----+  |       | PE1|=======\   /========| PE2|       |  +-----+
      |     |          |.......PW1.| \ / |............|       |  |     |
      | CE1 |----------|    |      |  X  |       |    |----------| CE2 |
      |     |  |       |.......PW2.| / \ |............|       |  |     |
      +-----+  |       |    |=======/   \========|    |       |  +-----+
            ^  |       +----+   ^  +------+      +----+       |  ^
            |  |      Provider  |     ^         Provider      |  |
            |  |       Edge 1   |     |          Edge 2       |  |
     Customer  |                |  P Router                   | Customer
      Edge 1   |             TE LSP                           |  Edge 2
               |                                              |
               |                                              |
              AC                                             AC
```

Figure 1: MPLS-TP Architecture (Single Segment PW) from RFC5921

A Pseudowire (PW) is configured between an ingress attachment circuit on a head-end switch (Provider Edge 1, PE1) and an egress attachment circuit on a tail-end node (Provider Edge 2, PE2). For a complete service, a PW must be configured in each direction.

2. The Reference Topology

Relevant components from the above architecture diagram are shown in the following reference topology.

```
                PW                                           PW
           End Service                                  End Service
    <-------------------------- Pseudo Wire -------------------------->
    |                                                                 |
    |                   +------------------------+                    |
    |                   | Management Application |                    |
    |                   +------------A-----------+                    |
    |                                |                                |
    |                        +-------V-------+                        |
    |        +---------------| OF Controller |---------------+        |
    |        |               +---------------+               |        |
    V        |                                               |        V
    +--------|--------------+                 +--------------|--------+
    |        | +----------+ |                 | +----------+ |        |
    |        | |OAM Engine| |                 | |OAM Engine| |        |
    |        | +----A-----+ |                 | +-----A----+ |        |
    |        |      |       |                 |       |      |        |
    | +------V------V-----+ |                 | +-----V------V------+ |
    | |                   | | Primary T-LSP   | |                   | |
    | |              [VP1]***************************[VP3]          | |
   -+-[AC1]               | |  Backup T-LSP   | |               [AC2]-+-
    | |              [VP2]***************************[VP4]          | |
    | |                   | |                 | |                   | |
    | | Virtual OF Switch | |                 | | Virtual OF Switch | |
    | +-------------------+ |                 | +-------------------+ |
    |    Head-end node      |                 |      Tail-end node    |
    +-----------------------+                 +-----------------------+
```

Figure 2: Reference topology

Pseudowires are configured from a network / provisioning Management Application that communicates with MPLS-TP nodes through an OpenFlow Controller (OF Controller). The Management Application configures an end-to-end Pseudo-wire ([PW1]) between Attachment Circuit [AC1] on the headend node and Attachment Circuit [AC2] on the tail-end node. At the head-end, all traffic coming from an input port (Attachment Circuit [AC1]) is switched onto the PW, and at the tail-end all traffic coming from the Pseudo-wire is switched to an output port (Attachment Circuit [AC2]). The Pseudo-wire is assigned a PW Label [PWL1]. The Management Application configures both packet forwarding and OAM function related to pseudowires.

In the reference topology, the head-end and tail-end nodes are

connected via a pair of transport LSPs - a primary transport LSP and a secondary transport LSP. Configuration and setup of transport the LSPs is outside the scope of this document. Assume that the head-end node pushes the appropriate transport LSP label onto packets entering a transport LSP, and the tail-end pops the transport LSP label from packets exiting from a transport LSP.

2.1. The MPLS-TP Node

An MPLS-TP Node has two major programmable components: a Virtual OF Switch and an OAM Engine, shown in the following figure.

```
                                +------------+
                                | Controller |
                                +------------+
                                      |
                                      V
                 +-------------------------------------------+
                 |           +-------------------+           |
               --+---->[In]->|                   |->[Out]----+--
                 |           |                   |           |
               --+---->[In]->| Virtual OF Switch |->[Out]----+--
                 |           |                   |           |
                 |  +->[In]->|                   |->[Out]-+  |
                 |  |        +-------------------+        |  |
                 |  |          +---------------+          |  |
                 |  +----------|   OAM Engine  |<---------+  |
                 |             +---------------+             |
                 |               MPLS-TP Node                |
                 +-------------------------------------------+
```

Figure 3: MPLS-TP Node Components

The Virtual OF Switch performs packet switching. It is controlled by an external controller, which in turn is driven by an MPLS-TP Management Application.

The OAM Engine generates and receives&processes OAM packets. It can perform OAM functions for both Pseudowires and Transport LSPs. At the head-end node, the OAM Engine is connected to a Virtual OF Switch input port. OAM packets are switched through the Virtual OF Switch either onto either Pseudowires or onto the transport LSPs. At the tail-end node, the OAM Engine is connected to a Virtual OF Switch output port. OAM packets are switched either from transport LSPs or Pseudowires to the OAM Engine. The tail-end node OAM Engine detects failure conditions. The head-end OAM Engine performs corrective actions.

2.1.1. The Virtual OF Switch

The Virtual OF switch is comprised of a single flow table (Flow Table 1) and a single group table. The Virtual OF Switch is shown in the following figure.

```
                            +------------+
                            | Controller |
                            +------------+
                                  |
                                  V
               +------------------------------------------+
               |          +-------+   +-------+           |
             --+--->[In]->|       |   |       |->[Out]---+--
               |          | Flow  |   | Group |           |
             --+--->[In]->| Table |-->| Table |->[Out]---+--
               |          |   1   |   |       |           |
             --+--->[In]->|       |   |       |->[Out]---+--
               |          +-------+   +-------+           |
               |             Virtual OF Switch            |
               +------------------------------------------+
```

Figure 4: The Virtual OF Switch

Although the Virtual OF switch is the same a both the head-end and tail-end nodes, the group table is only used at the head-end node, where a Fast Failover group is set up for each pair of ports that correspond to the primary-backup Transport LSP pair. At the tail-end node, only flows are set up.

Transport LSPs are identified as ports to the OF controller. for the Reference Topology in Figure 2, the primary transport LSP is identified to OpenFlow as Virtual Port [VP1] and the backup transport LSP is identified as Virtual Port [VP2]. At the head-end switch in the Reference Topology a Fast Failover group [PG1] is set up for ports [VP1] and [VP2].

The Virtual OF Switch MUST support the following OFPXMC_OPENFLOW_BASIC match fields ([OF-1.3.0], Section A.2.3.7):

o Match on Switch Input Port (OFPXMT_OFB_IN_PORT)

o Match on MPLS Label (OFPXMT_OFB_MPLS_LABEL)

o Match on MPLS BoS bit (OFPXMT_OFP_MPLS_BOS)

The Virtual OF Switch must support the following OF actions ([OF-1.3.0], Section A.2.5):

- o Output to switch port (OFPAT_OUTPUT)
- o Push a new MPLS tag (OFPAT_PUSH_MPLS)
- o Pop the outer MPLS tag (OFPAT_POP_MPLS)
- o Apply group (OFPAT_GROUP)

2.1.2. The OAM Engine

The liveness of Transport LSPs and PWs is monitored by OAM. The desired model is to have an OAM engine residing locally on the switch. At the ingress to a PW the OAM Engine will inject OAM packets into the PW data path. At the egress of a PW the switch will detect OAM packets (performing a flow-match operation) and punt OAM packets to the OAM Engine, which will evaluate them, and if need be perform corrective action and/or produce notifications.

The OAM function for Transport LSPs is out of scope for this revision of this document. However, PW-OAM mechanisms described in this document are also applicable to Transport LSP-OAM.

In addition to generating and processing OAM packets, the OAM Engine participates liveness monitoring function ([OF-1.3.0], Section 6.6) for virtual port Fast Failover groups at the head-end switch. When the OAM Engine detects a PW failure, it triggers the Virtual OF Switch to move traffic from the primary transport LSP's virtual port [VP1] to the backup transport LSP's virtual port [VP2]. The OAM Engine's liveness monitoring function is described in more detail in [OF-1.3.0].

Note that in addition to the OAM Engine, the Virtual OF Switch MAY use other liveness monitoring mechanisms for the virtual port Fast Failover groups, which are out of scope of this document.

3. PW Configuration

3.1. Configuration Messages

The Controller uses OpenFlow protocol messages defined in [OF-1.3.0] to configure transport pseudo-wires. The Flow Modification message and the Group Modification message types are used. To configure a PW at the head-end node, the Controller uses a sequence of a Group Modification message followed by a Flow Modification message. At the tail-end node, the Controller uses Flow Modification messages only.

The message formats in this document are specified using Routing

Backus-Naur Format (RBNF) encoding as specified in [RFC5511].

3.1.1. The Flow Modification Message

The Flow Modification message - 'Modify Flow Entry' - is defined in [OF-1.3.0], Section A.3.4.1 as follows:

```
    <ofp-flow-mod> ::= <ofp-header>
                       <COOKIE>
                       <COOKIE_MASK>
                       <TABLE_ID>
                       <COMMAND>
                       <IDLE_TIMEOUT>
                       <HARD_TIMEOUT>
                       <PRIORITY>
                       <BUFFER_ID>
                       <OUT_PORT>
                       <OUT_GROUP>
                       <FLAGS>
                       <ofp-match>
                       <instructions>

    <ofp-header> ::= <VERSION>
                     <OFP_MSG_TYPE>
                     <LENGTH>
                     <XID>

    <ofp-match> ::= <MATCH_TYPE>
                    <MATCH_LENGTH>
                    <oxm_fields>

    <oxm-fields> ::= <oxm-tlv> [<oxm-fields>]

    <oxm-tlv> ::= <OXM_CLASS>
                  <OXM_FIELD>
                  <OXM_HASHMASK>
                  <OXM_LENGTH>
                  <PAYLOAD>

    <instructions> ::= <instruction> [<instructions>]

    <instruction> ::= ( <ofp-instruction-actions> |
                        <ofp-instruction-write-metadata> |
                        <ofp-instruction-goto-table> |
                        <ofp-instruction-meter> )

    <ofp-instruction-actions> ::= <TYPE>
                                  <LEN>
```

```
                                     <PAD>
                                     <actions>

       <actions> ::= <ofp-action> [<actions>]

       <ofp-action> ::= (<ofp-action-output> |
                         <ofp-action-group> |
                         <ofp-action-push-mpls> |
                         <ofp-action-pop-mpls> | ...)

       <ofp-action-group> ::= <TYPE>
                              <LEN>
                              <GROUP_ID>

       <ofp-action-output> ::= <TYPE>
                               <LEN>
                               <PORT>
                               <MAX_LEN>

       <ofp-action-push-mpls> ::= <TYPE>
                                  <LEN>
                                  <ETHERTYPE>
                                  <MPLS_HEADER>


       <ofp-action-pop-mpls> ::= <TYPE>
                                 <LEN>
                                 <ETHERTYPE>
```

Figure 5: The 'Modify Flow Entry' message

Note that not all action types defined in [OF-1.3.0] for <ofp-action> are listed in Figure 5.

3.1.2. The Group Modification Message

The Group Modification message - 'Modify Group Entry' - is defined in [OF-1.3.0], Section A.3.4.2 as follows:

```
<ofp-group-mod> ::= <ofp-header>
                    <COMMAND>
                    <GROUP_MSG_TYPE>
                    <GROUP_ID>
                    <buckets>

<ofp-header> ::= <VERSION>
                 <OFP_MSG_TYPE>
                 <LENGTH>
                 <XID>

<buckets> ::= <ofp-bucket> [<buckets>]

<ofp-bucket> ::= <LEN>
                 <WEIGHT>
                 <WATCH_PORT>
                 <WATCH_GROUP>
                 <actions>

<actions> ::= <ofp-action> [<actions>]

<ofp-action> ::= (<ofp-action-output> | <ofp-action-group> | ...)

<ofp-action-output> ::= <TYPE>
                        <LEN>
                        <PORT>
                        <MAX_LEN>
```

Figure 6: The 'Modify Group Entry' message

3.2. PW Head-End Node Configuration

Consider the reference topology in Figure 2. The Management Application will configure a cross-connect between the Attachment Circuit [AC1] and the virtual port pair {[VP1], [VP2]} joined in the Fast Failover group [PG1]. The cross-connect determines that traffic from Port AC will be switched to Group PG1. The internal mechanism in Group PG1 (outside the scope of this document) will determine whether traffic will go out on Port [VP1] (the primary transport LSP) or on Port [VP2] (the backup transport LSP).

The Management Application uses the following message sequence to create the cross-connect:

1. The 'Modify Group Entry' message, defined in Figure 6 creates or modifies an entry in the Group Table. Each entry in the Group Table corresponds to a pair of virtual ports that correspond to a pair of primary / backup transport LSPs. This entry states that

as long as Port [VP1] is alive, traffic coming to group [PG1] will go out on [VP1]. If Port [VP1] is not alive AND Port [VP2] is alive, then traffic coming to group [PG1] will go out on Port [VP2]. If neither of the ports are alive, traffic will be dropped. Note that it's up to the switch to determine that a given port is alive - and it can use any mechanism that it wants to do that.

2. The 'Modify Flow Entry' message, defined in Figure 5, adds an entry to Flow Table 1 for a flow that matches traffic from Input Port [AC1]. The actions for the flow are 1.) Push the PW MPLS header on the packet, and 2.) Forward the packet to Group [PG1], which was setup in Step 1).

The following sections describe in details each message.

3.2.1. 'Modify Group Entry' Message Details

The fields in the 'Modify Group Entry' message are set as follows:>

'Modify Group Entry' message: <COMMAND> is set to 'OFPGC_ADD' or 'OFPGC_MODIFY', <GROUP_MSG_TYPE> is set to 'OFPGT_FF'(fast failover group) and <GROUP_ID> is set to the identifier of the Fast Failover group that was setup for the primary and secondary transport LSPs - [PG1].

OpenFlow Header (ofp-header): <VERSION> is set to 4, <OFP_MSG_TYPE> is set to 'OFPT_GROUP_MOD'.

Buckets: there are two action buckets - Bucket1 and Bucket2:

Bucket1 is associated with the virtual port corresponding to the primary transport LSP, and its fields are set as follows. <WEIGHT> is set to 1, <WATCH_PORT> is set to [VP1], <WATCH_GROUP> is set to 'OFPG_ANY'. <action-list> contains a single item - an <ofp-action-output> to Virtual Port [VP1].

Bucket2 is associated with the virtual port corresponding to the backup transport LSP, and its fields are set as follows. <WEIGHT> is set to 10, <WATCH_PORT> is set to [VP2], <WATCH_GROUP> is set to 'OFPG_ANY'. <action-list> contains a single item - an <ofp-action-output> to Virtual Port [VP2].

3.2.2. 'Modify Flow Entry' Message Details

The fields in the Modify Flow Entry' message are set as follows:

'Modify Flow Entry' message: The controller MUST set the value of <TABLE_ID> to '1', the value of <COMMAND> 'OFPFC_MODIFY_STRICT', the value of <BUFFER_ID> to 'OFP_NO_BUFFER', the value of <OUT_PORT> to 'OFPP_ANY', and the value of <OUT_GROUP> to 'OFPG_ANY'. The Controller SHOULD set the values of all other atomic fields to appropriate values as required by the operation of the configuration protocol. It is recommended that the 'IDLE_TIMEOUT' and 'HARD_TIMEOUT' fields are set to 0 for persistant PW configurations

OpenFlow Header (ofp-header): <VERSION> is set to 4, <OFP_MSG_TYPE> is set to 'OFPT_GROUP_MOD'.

Ofp-Match: The Controller MUST set the <MATCH_TYPE> field to 'OFPMT_OXM' and include a single <oxm-tlv> with the <OXM_CLASS> field set to 'OFPXMC_OPENFLOW_BASIC', the <OXM_FIELD> field set to 'OFPXMT_OFB_IN_PORT', the <OXM_HASHMASK> field set to '0', and the <PAYLOAD> field set to [AC1].

Instructions: The Controller MUST set the type field <TYPE> to 'OFPIT_APPLY_ACTIONS' and include the following action list:

Push MPLS Header: the value of <TYPE> set to 'OFPAT_PUSH_LABEL'; the value of <ETHERTYPE> set to MPLS Unicast; the value of <MPLS_HEADER> set as follows: Label=[PWL1], TTL=1, TC=???, S=1.

Group: the value of <TYPE> set to 'OFPAT_GROUP' and the value of <GROUP_ID> set to [PG1].

3.3. PW Tail-End Node Configuration

Consider the reference topology in Figure 2. The Management Application will configure two cross-connects: one cross-connect between the primary Transport LSP's virtual port ([VP3]) and the Attachment Circuit [AC2], and one between the primary transport LSP's virtual port ([VP3]) and the Attachment Circuit [AC2]. Under normal circumstances, traffic will arrive at the primary Transport LSP's Virtual Port [VP3]. When the primary Transport LSP is not available and the backup Transport LSP is ok, traffic will arrive at the backup Transport LSP's Virtual Port [VP4].

Note that the cross-connect between the backup Transport LSP's Virtual Port [VP4] and the Attachment Circuit [AC2] can be programmed along with the cross-connect between the primary Transport LSP's Virtual Port [VP3] and the Attachment Circuit [AC2], or at the time when the primary Transport LSP's Virtual Port [VP3] goes down.

The cross-connects between the primary Transport LSP's Virtual Port

[VP3] and the Attachment Circuit [AC2], and between the backup transport LSP's Virtual Port [VP4] and the Attachment Circuit [AC2] are programmed by sending 'Modify Flow Entry' messages to the switch. Programming details are described in the following section.

3.3.1. 'Modify Flow Entry' Message Details

The fields in the Modify Flow Entry' messages are set as follows:

'Modify Flow Entry' message: The controller MUST set the value of <TABLE_ID> to '1', the value of <COMMAND> 'OFPFC_MODIFY_STRICT', the value of <BUFFER_ID> to 'OFP_NO_BUFFER', the value of <OUT_PORT> to 'OFPP_ANY', and the value of <OUT_GROUP> to 'OFPG_ANY'. The Controller SHOULD set the values of all other atomic fields to appropriate values as required by the operation of the configuration protocol. It is recommended that the 'IDLE_TIMEOUT' and 'HARD_TIMEOUT' fields are set to 0 for persistant PW configurations

OpenFlow Header (ofp-header): <VERSION> is set to 4, <OFP_MSG_TYPE> is set to 'OFPT_GROUP_MOD'.

Ofp-Match: The Controller MUST set the <MATCH_TYPE> field to 'OFPMT_OXM' and include the following <oxm-tlv> match list:

Match Input Port: the value of the <OXM_CLASS> field set to 'OFPXMC_OPENFLOW_BASIC'; the value of the <OXM_FIELD> field set to 'OFPXMT_OFB_IN_PORT'; the value of the <OXM_HASHMASK> field set to '0' and the value <PAYLOAD> field set to [VP3] (for the primary Transport LSP) or [VP4] (for the backup Transport LSP).

Match MPLS Label: the value of the <OXM_CLASS> field set to 'OFPXMC_OPENFLOW_BASIC'; the value of the <OXM_FIELD> field set to 'OFPXMT_OFB_MPLS_LABEL'; the value of the <OXM_HASHMASK> field set to '0' and the value <PAYLOAD> field set to [PWL1].

Instructions: The Controller MUST set the type field <TYPE> to 'OFPIT_APPLY_ACTIONS' and include the following action list:

Pop MPLS Header: the value of <TYPE> set to 'OFPAT_POP_MPLS' and the value of <ETHERTYPE> set to MPLS Unicast.

Output: the value of <TYPE> set to 'OFPAT_OUTPUT' and the value of <PORT> set to [AC2].

4. PW OAM Considerations

4.1. OAM Overview

OAM for MPLS-TP is an important consideration and needs to be addressed in a scalable manner and needs to function with the same performance available today. Centralization of control or digestion of OAM messages, where they are redirected back to a central controller will introduce delay. Therefore the goal is to drive OAM setup for PWs using messages from the Controller to the Switch. The functions of the OAM, for example OAM packet generation, error detection, action/notification will therefore still reside locally on the switch.

We are using [RFC6423] as a reference for unified OAM for MPLS-TP, in particular Section 3. which includes provision for GAL with PW in MPLS-TP.

4.2. OAM Engine Mechanism

If OAM is required on a particular PW it requires only a small number of OF actions to enable OAM.

- o The head-end OAM Engine is programmed to generate OAM packets for the PW. The header for these packets will composed of at least 2 labels, the first being the PW label for which the OAM is being generated, and the following label being the GAL label (13). The contents of the G.ach packet are out of scope for this draft and will be down to the individual OAM implementation

- o The head-end Virtual OF switch is programmed to switch the OAM packets generated by the OAM Engine to the respective destination PW. As the PW label is being placed on the OAM packet, we can easily match on this and forward the OAM packet down the correct PW to ensure it follows the same data path.

- o The tail-end Virtual OF switch is programmed to switch OAM packets received from the PW to the OAM Engine. The PW label remains intact when the OAM packet is punted to the OAM Engine to allow for identification of the PW from which the OAM came. The match rule will look for both the PW label and the GAL label to correctly identify an OAM packet

- o The tail-end OAM Engine is programmed to receive OAM packets and to detect failures.

OAM mechanisms that can be implemented by the OAM Engine are out of scope for this revision of the document. For example, the OAM Engine

can also implement BFD (Echo) Mode [RFC5880] where echo packets are returned via the remote forwarding plane, which can be done using an OF match rule.

4.3. OAM and S-Bit considerations

In order to ensure that the switch can identify the last label in the stack the S bit needs to be set on the label which will be at the bottom of the stack. The OAM Engine will be required to set the S bit on the GAL label (13) to ensure that the subsequent G-ach packet is treated correctly. By using OF actions to move all OAM Engine packets into the PW we ensure that not only all types of OAM are supported transparently, but also that the S bit is correctly set.

5. OAM and OF Controller Actions

The OF Switch will need to have a number of actions programed into the forwarding table to ensure that OAM packets are directed to and/or from the OAM Engine in order to achieve the desired operations detailed in Section 4.2.

6. OAM Head-End Actions

Below is an example of a rule to push all OAM packets coming from the OAM Engine related to a particular PW down that PWs path.

```
       <ofp-match> ::= <MATCH_TYPE>    ; ='OFPMT_OXM'
                       <MATCH_LENGTH>
                       <oxm_fields>

       <oxm-fields> ::= <oxm-tlv> [<oxm-fields>]

     <oxm-tlv> ::= <OXM_CLASS>
                     <oxm_VALUE>                  ; = '0x8847'
                     <OXM_OF_ETH_TYPE>            ; = 'MPLS'
                     <OXM_FIELD>
                     <OXM_HASHMASK>
                     <OXM_LENGTH>                 ; = ''
                     <OXM_OF_MPLS_LABEL>          ; = '100
       <oxm-tlv> ::= <OXM_CLASS>
                     <oxm_VALUE>                  ; = '0x8847'
                     <OXM_OF_ETH_TYPE>            ; = 'MPLS'
                     <OXM_FIELD>
                     <OXM_HASHMASK>
                     <OXM_LENGTH>                 ; = ''
                     <OXM_OF_MPLS_LABEL>          ; = '13'

          <actions> ::= <ofp-action> [<actions>]

          <ofp-action> ::= (<ofp-action-output> |
                            <ofp-action-group> |
                            <ofp-action-push-mpls> |
                            <ofp-action-pop-mpls> | ...)

          <ofp-action-group> ::= <TYPE>           ; ='OFPAT_GROUP'
                                 <LEN>
                                 <GROUP_ID>       ; =[PG1]
```

Figure 7: Head-End OAM Match

6.1. OAM Tail-End Action

Below is an example of a rule to push all OAM packets, received on the tail-end PW. Note that neither the PW label or the GAL label are removed, as they are using to identify the PW they are performing OAM for.

```
       <ofp-match> ::= <MATCH_TYPE>    ; ='OFPMT_OXM'
                       <MATCH_LENGTH>
                       <oxm_fields>

       <oxm-fields> ::= <oxm-tlv> [<oxm-fields>]

     <oxm-tlv> ::= <OXM_CLASS>
                     <oxm_VALUE>                 ; = '0x8847'
                     <OXM_OF_ETH_TYPE>           ; = 'MPLS'
                     <OXM_FIELD>
                     <OXM_HASHMASK>
                     <OXM_LENGTH>                ; = ''
                     <OXM_OF_MPLS_LABEL>         ; = '100'
       <oxm-tlv> ::= <OXM_CLASS>
                     <oxm_VALUE>                 ; = '0x8847'
                     <OXM_OF_ETH_TYPE>           ; = 'MPLS'
                     <OXM_FIELD>
                     <OXM_HASHMASK>
                     <OXM_LENGTH>                ; = ''
                     <OXM_OF_MPLS_LABEL>         ; = '13'

        <ofp-action-output> ::= <TYPE>          ; ='OFPAT_OUTPUT'
                                <LEN>
                                <PORT>          ; ='OAM CONTROLLER'
                                <MAX_LEN>
```

Figure 8: Tail-end OAM Match

7. IANA Considerations

This document does not introduce any IANA requirements.

8. Security Considerations

Procedures described in this document do not change the OpenFlow protocol security model described in [OF-1.3.0], Section 6.3.

A secure communications channel SHOULD be set up between the controller and the MPLS-TP node.

9. Acknowledgements

The authors would like to thank Giles Heron, Dave Ward, Frank Brockners and Dan Frost for their reviews and comments.

10. Normative References

[OF-1.3.0]
Open Networking Foundation, "OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04)", April 16, 2012.

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

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5921] Bocci, M., Bryant, S., Frost, D., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, July 2010.

[RFC6423] Li, H., Martini, L., He, J., and F. Huang, "Using the Generic Associated Channel Label for Pseudowire in the MPLS Transport Profile (MPLS-TP)", RFC 6423, November 2011.

Authors' Addresses

Jan Medved
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: jmedved@cisco.com

Andrew McLachlan
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: amclachl@cisco.com

David Meyer
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: dmm@cisco.com