INTERNET DRAFT

Internet Engineering Task Force
Document:
draft-ietf-l2vpn-vpls-pim-snooping-00.txt

July 2006
Category: Informational
Expires: January 2007

Venu Hemige
Alcatel
Yetik Serbest
AT&T
Ray Qiu
Suresh Boddapati
Alcatel

PIM Snooping over VPLS

Status of this memo

Abstract

In Virtual Private LAN Service (VPLS), as also in IEEE Bridged Networks, the switches simply flood multicast traffic on all ports in the LAN by default. IGMP Snooping is commonly deployed to ensure multicast traffic is not forwarded on ports without IGMP receivers. The procedures and recommendations for IGMP Snooping are defined in [IGMP-SNOOP]. But when any protocol other than IGMP is used, the common practice is to simply flood multicast traffic to all ports. PIM-SM, PIM-SSM, PIM-BIDIR are widely deployed routing protocols. PIM Snooping procedures are important to restrict multicast traffic to only the routers interested in receiving such traffic.

While most of the PIM Snooping procedures defined here also apply to IEEE Bridged Networks, VPLS demands certain special procedures due to the split-horizon rules that require the Provider Edge (PE) devices to co-operate. This document describes the procedures and recommendations for PIM-Snooping in VPLS to facilitate replication to only those ports behind which there are interested PIM routers and/or IGMP hosts.

This document also describes procedures for PIM Proxy. PIM Proxy is required on PEs for VPLS Multicast to work correctly when Join suppression is enabled in the VPLS. PIM Proxy also helps scale VPLS Multicast much better than just PIM Snooping.

Conventions used in this document

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

Table of Contents

1. Introduction

In Virtual Private LAN Service (VPLS), the Provider Edge (PE) devices provide a logical interconnect such that Customer Edge (CE) devices belonging to a specific VPLS instance appear to be connected by a single LAN. Forwarding information base for particular VPLS instance is populated dynamically by source MAC address learning. This is a straightforward solution to support unicast traffic, with reasonable flooding for unicast unknown traffic. Since a VPLS provides LAN emulation for IEEE bridges as wells as for routers, the unicast and multicast traffic need to follow the same path for layer-2 protocols to work properly. As such, multicast traffic is treated as broadcast traffic and is flooded to every site in the VPLS instance.

VPLS solutions (i.e., [VPLS-LDP] and [VPLS-BGP]) perform replication for multicast traffic at the ingress PE devices. When replicated at the ingress PE, multicast traffic wastes bandwidth when:

1. Multicast traffic is sent to sites with no members,
2. Pseudo wires to different sites go through a shared path, and
3. Multicast traffic is forwarded along a shortest path tree as opposed to the minimum cost spanning tree.

This document is addressing the first problem by IGMP and PIM snooping. Problems #2 and #3 are orthogonal to #1 and outside the scope of this document. The different mechanisms to tunnel IP multicast traffic in a VPLS from the ingress PE to the egress PEs are discussed in [VPLS-MCAST-TREES].

Using VPLS in conjunction with IGMP and/or PIM snooping has the following advantages:

- It improves VPLS to support IP multicast efficiently (not necessarily optimum, as there can still be bandwidth waste if traffic from a PE to other PE(s) is not forwarded along a minimum cost spanning tree.),
- It prevents sending multicast traffic to sites with no members.

Procedures for IGMP Snooping are specified in [IGMP-SNOOP]. This document describes the procedures for Protocol Independent Multicast (PIM) snooping over VPLS for efficient distribution of IP multicast traffic. It also describes the rules when both IGMP and PIM are active in a VPLS instance.

This document also describes procedures for PIM Proxy. PIM Proxy is required on PEs for VPLS Multicast to work correctly when Join suppression is enabled in the VPLS. PIM Proxy also helps scale VPLS Multicast much better than just PIM Snooping.

1.1. Assumptions

Since this draft describes the procedures for PIM Snooping and PIM Proxy, the draft assumes that the reader has a good understanding of the PIM protocols. The text in this draft is written in the same style as the PIM RFCs to help correlate the concepts and to make it easier to follow. In order to avoid replicating the text relating to PIM protocol handling here, this draft assumes that the user will infer such detail from the PIM RFC referenced in this document. Deviations in protocol handling specific to PIM Snooping and PIM Proxy are specified in this draft. There could be cross references into definitions of macros and procedures from the PIM RFCs.

1.2. Definitions

There are several definitions referenced in this document that are well described in the PIM RFCs [PIM-SM, PIM-BIDIR, PIM-DM].

The following definitions and abbreviations are used throughout this document:

- A port is defined as either an attachment circuit (AC) or a Pseudo-Wire (PW).
- When we say a PIM message is 'received' on a port, it means any one of the following:
  - o that a PIM Snooping switch snooped the PIM message.
  - o that a PIM message was received via LDP on a PW if LDP (as defined in [VPLS-MCAST-LDP]) is used for propogating multicast states among the PEs.

Abbreviations used in the document:

- S: IP Address of the Multicast Source.
- G: IP Address of the Multicast Group.
- N: Upstream Neighbor field in a Join/Prune/Graft message.
- Rport(X): Port on which neighbor X is learnt

Other definitions are explained in the sections where they are introduced.

2. Multicast Traffic over VPLS

In VPLS, if a PE receives a frame from an Attachment Circuit (AC) with no matching entry in the forwarding information base for that particular VPLS instance, it floods the frame to all other PEs (which are part of this VPLS instance) and to directly connected ACs (other than the one that the frame is received from). The flooding of a frame occurs when:

- The destination MAC address has not been learned,
- The destination MAC address is a broadcast address,
- The destination MAC address is a multicast address.

Malicious attacks (e.g., receiving unknown frames constantly) aside, the first situation is handled by VPLS solutions as long as destination MAC address can be learned. After that point on, the frames will not be flooded. A PE is REQUIRED to have safeguards, such as unknown unicast limiting and MAC table limiting, against malicious unknown unicast attacks.

There is no way around flooding broadcast frames. To prevent runaway broadcast traffic from adversely affecting the VPLS service and the SP network, a PE is REQUIRED to have tools to rate limit the broadcast traffic as well.

Similar to broadcast frames, multicast frames are flooded as well, as a PE cannot know where multicast members reside. Rate limiting multicast traffic, while possible, should be should be done carefully since several network control protocols relies on multicast. For one thing, layer-2 and layer-3 protocols utilize multicast for their operation. For instance, Bridge Protocol Data Units (BPDUs) use an IEEE assigned all bridges multicast MAC address, and OSPF is multicast to all OSPF routers multicast MAC address. If the rate-limiting of multicast traffic is not done properly, the customer network will experience instability and poor performance. For the other, it is not straightforward to determine the right rate limiting parameters for multicast.

A VPLS solution MUST NOT affect the operation of customer layer-2 protocols (e.g., BPDUs). Additionally, a VPLS solution MUST NOT affect the operation of layer-3 protocols.

In the following section, we describe procedures to constrain the flooding of IP multicast traffic in a VPLS.

2.1. Constraining of IP Multicast in a VPLS

For a PE in a VPLS (a layer-2 device) to constrain IP multicast traffic, it needs to be able to learn which CEs are interested in receiving multicast traffic for what flows.

The most obvious solution is to snoop IP multicast control traffic at the PEs. Snooping as a solution to constrain multicast traffic makes sense under the following circumstances:

- The CE-CE protocol the PEs snoop is a popular and widely deployed protocol.
- It does not require any changes on the CEs and it should be completely transparent to the CEs.

Using VPLS in conjunction with IGMP and/or PIM snooping has the following advantages:

- It improves VPLS to support IP multicast efficiently (not necessarily optimum, as there can still be bandwidth waste if traffic from a PE to other PE(s) is not forwarded along a minimum cost spanning tree.),
- It prevents sending multicast traffic to sites with no members.

Other routing protocols such as DVMRP or MOSPF are outside the scope of this document.

In the following sub-sections, we provide some guidelines for the implementation of PIM snooping in VPLS. Snooping techniques need to be employed on ACs at the downstream PEs. Snooping techniques can also be employed on PWs at the upstream PEs. This may work well for small to medium scale deployments. However, if there are a large number of VPLS instances with a large number of PEs per instances, then the amount of snooping required at the upstream PEs can overwhelm the upstream PEs. In [VPLS-MCAST-LDP] and [VPLS-MCAST-BGP], procedures are defined to exchange multicast membership information between the PEs using LDP or BGP. Using a reliable mechanism like LDP or BGP allows the upstream PEs to eliminate the requirement to snoop on PWs. It also eliminates the need to refresh multicast states on the upstream PEs.

This document also describes the guidelines for PIM Proxy in VPLS. The specifications in this document could be used for either PIM Snooping or PIM Proxy. The PIM Proxy solution is described in section 3.6. Differences that need to be observed while implementing one or the other and recommendations on which method to employ in different scenarios are noted in section 2.4.

2.2. IPv6 Considerations

In VPLS, PEs forward Ethernet frames received from CEs and as such are agnostic of the layer-3 protocol used by the CEs. However, as an IGMP and PIM snooping switch, the PE would have to look deeper into the IP and IGMP/PIM packets and build snooping state based on that. The PIM Protocol specifications handle both IPv4 and IPv6. The specification for PIM Snooping in this draft can be applied to both IPv4 and IPv6 payloads.

2.3. PIM-SM (*,*,RP) Considerations

This draft does not address (*,*,RP) states in the VPLS network. Although [PIM-SM] specifies that routers MUST support (*,*,RP) states, there are very few implementations that actually support it in actual deployments. Given the complexity of supporting (*,*,RP) states and knowing that there is little to no use to supporting it, this draft omits the specification relating to (*,*,RP) support.

2.4. PIM Snooping vs PIM Proxy

PIM Snooping switches simply snoop on PIM packets as they are being forwarded in the VPLS. As such it truly provides transparent LAN services since no customer packets are modified or consumed or new packets introduced in the VPLS. It is also slightly simpler to implement than PIM Proxy. However for PIM Snooping to work correctly, it is a requirement that CE routers MUST disable Join suppression in the VPLS.

Given that a large number of existing CE deployments do not support disabling of Join suppression and given the operational complexity for a provider to manage disabling of Join suppression in the VPLS, it becomes a difficult solution to deploy. Another disadvantage of PIM Snooping as a solution is that it does not scale as well as PIM Proxy. If there are a large number of CEs in a VPLS, then every CE will see every other CE's Join/Prune messages.

PIM Proxy on the PEs has the advantage that it does not require Join suppression to be disabled in the VPLS. Multicast as a VPLS service can be very easily be provided without requiring any changes on the CE routers. It also helps scale VPLS Multicast very well since the PEs intelligently forward only one Join/Prune message for a given flow and only to the upstream CE.

PIM Proxy as a solution however loses the transparency argument since Join/Prunes could get modified or even consumed at a PE. Also, new packets could get introduced in the VPLS. However, this loss of transparency is limited to PIM control packets. It is in the interest of optimizing multicast in the VPLS and helping a VPLS network scale much better. Data traffic will still be completely transparent.

Both PIM Snooping and PIM Proxy procedures can be used in conjunction with [VPLS-MCAST-LDP] for propogating multicast states among the PEs. If [VPLS-MCAST-LDP] is used for propogating multicast states among the PEs, then both PIM Snooping and PIM Proxy switches do not process any PIM packets arriving on a PW.

2.4.1. Differences between PIM Snooping and PIM Proxy

For PIM-SM and PIM-BIDIR, a PIM Snooping/Proxy Switch only needs to examine PIM Hello and Join/Prune messages. PIM Proxy for PIM-DM is for future study and is not currently specified in this draft.

The proxy proposal is to perform proxy of only the Join/Prune messages while snooping Hello messages. Details on the PIM Proxy solution are discussed in section 3.6. This section is presented here to say that most of the procedures to follow (unless explicitly specified) are common to both PIM Snooping and PIM Proxy.

Differences between a PIM Snooping switch and a PIM Proxy switch can be summarized as the following:

PIM Snooping	PIM Proxy
1. PIM Snooping switches snoop Hello and Join/Prune messages while they are transparently flooded in the VPLS.	1. PIM Proxy switches also snoop PIM Hello messages while they are transparently flooded in the VPLS. But they consume PIM Join/Prune messages and do not flood them as is in the VPLS.
2. PIM Snooping switches do not originate any PIM packets. They may however originate PIM messages to be sent via LDP on PWs.	2. PIM Proxy switches may originate new or modified PIM packets.

Other than the above simple differences, most of the procedures are common to PIM Snooping and PIM Proxy. There are additional simplifications to PIM Snooping that can be made if [VPLS-MCAST-LDP] is not used for PE-PE communication, but otherwise the procedures for PIM Snooping and PIM Proxy are mostly the same. In the text to follow, we describe the text as procedures for PIM Snooping. Unless otherwise specified, such procedures apply to PIM Proxy as well.

3. PIM Snooping for VPLS

IGMP snooping procedures described in [IGMP-SNOOP] provide efficient delivery of IP multicast traffic in a given VPLS service when end stations are connected to the VPLS. However, when VPLS is offered as a WAN service it is likely that the CE devices are routers and would run PIM between them. To provide efficient IP multicasting in such cases, it is necessary that the PE routers offering the VPLS service do PIM snooping.

PIM is a multicast routing protocol, which runs exclusively between routers. PIM shares many of the common characteristics of a routing protocol, such as discovery messages (e.g., neighbor discovery using Hello messages), topology information (e.g., multicast tree), and error detection and notification (e.g., dead timer and designated router election). On the other hand, PIM does not participate in any kind of exchange of databases, as it uses the unicast routing table to provide reverse path information for building multicast trees. There are a few variants of PIM. In PIM-DM ([PIM-DM]), multicast data is pushed towards the members similar to broadcast mechanism. PIM-DM constructs a separate delivery tree for each multicast group. As opposed to PIM-DM, other PIM flavors (PIM-SM [PIM-SM], PIM-SSM [PIM-SSM], and PIM-BIDIR [PIM-BIDIR]) invoke a pull methodology instead of push technique.

PIM routers periodically exchange Hello messages to discover and maintain stateful sessions with neighbors. After neighbors are discovered, PIM routers can signal their intentions to join or prune specific multicast groups. This is accomplished by having downstream routers send an explicit Join/Prune message (for the sake of generalization, consider Graft messages for PIM-DM as Join messages) to the upstream routers. The Join/Prune message can be group specific (*,G) or group and source specific (S,G).

In PIM snooping, a PE snoops on the PIM message exchange between routers, and builds its multicast states.

Based on the multicast states, it forwards IP multicast traffic accordingly to avoid unnecessary flooding.

In the following sub-sections, snooping mechanisms for each variety of PIM are specified.

3.1. General Rules for PIM Snooping in VPLS

The following rules for the correct operation of IGMP/PIM snooping MUST be followed.

- IGMP messages, PIM messages and multicast data traffic forwarded by PEs MUST follow the split-horizon rule for mesh

PWs as defined in [VPLS-LDP].

- IGMP/PIM snooping states in a PE MUST be per VPLS instance.
- Multicast traffic MUST be replicated per PW and AC basis, i.e., even if there are more than one PIM neighbor behind a PW/AC, only one replication MUST be sent to that PW/AC.

3.1.1. Snooping PIM Packets

PIM-SM and PIM-BIDIR snooping PEs need to snoop on just the PIM Hello and PIM Join/Prune messages to build its multicast states.

- PIM-DM snooping PEs have to also snoop on PIM Graft and PIM State Refresh messages.

3.2. Discovering PIM Routers

A PIM Snooping PE MUST snoop on PIM Hellos received on ACs and PWs. PIM Hellos are used by the snooping switch to discover PIM routers and their characteristics.

For each neighbor discovered by a PE, it includes an entry in the PIM Neighbor Database with the following fields:

- Layer 2 encapsulation for the Router sending the PIM Hello.
- IP Address and address family of the Router sending the PIM Hello.
- Port (AC / PW) on which the PIM Hello was received.
- Hello TLVs

The PE should be able to interpret and act on Hello TLVs currently defined in the PIM RFCs. The TLVs of particular interest in this document are:

- Hello-Hold-Time
- Tracking Support
- DR Priority

Please refer to [PIM-SM] for a list of the Hello TLVs.

When a PIM Hello is received, the PE MUST reset the neighbor-expiry-timer to Hello-Hold-Time. If a PE does not receive a Hello message from a router within Hello-Hold-Time, the PE MUST remove that neighbor from its PIM Neighbor Database. If a PE receives a Hello message from a router with Hello-Hold-Time value set to zero, the PE MUST remove that router from the PIM snooping state immediately.

From the PIM Neighbor Database, a PE MUST be able to use the procedures defined in [PIM-SM] to identify the Designated Router in

the VPLS instance. It should also be able to determine if Tracking Support is active in the VPLS instance.

3.3. PIM-SM and PIM-SSM

The key characteristic of PIM-SM and PIM-SSM is explicit join behavior. In this model, multicast traffic is only forwarded to locations that specifically request it. The root node of a tree is the Rendezvous Point (RP) in case of a shared tree (PIM-SM only) or the first hop router that is directly connected to the multicast source in the case of a shortest path tree. All the procedures described in this section apply to both PIM-SM and PIM-SSM, except for the fact that there is no (*,G) state in PIM-SSM.

The procedures to discover PIM-SM routers in a VPLS instance are as described in section 3.2.

3.3.1. Building PIM-SM Snooping States

PIM-SM and PIM-SSM Snooping states are built by snooping on the PIM-SM Join/Prune messages received on AC/PWs.
The downstream state machine of a PIM-SM snooping switch very closely resembles the downstream state machine of PIM-SM routers. The downstream state consists of:

Per downstream (Port, *, G):

- DownstreamJPState: One of { "NoInfo" (NI), "Join" (J), "Prune Pending" (PP) }

Per downstream (Port, *, G, N):

- Prune Pending Timer (PPT(N))
- Join Expiry Timer (ET(N))

Per downstream (Port, S, G):

- DownstreamJPState: One of { "NoInfo" (NI), "Join" (J), "Prune Pending" (PP) }

Per downstream (Port, S, G, N):

- Prune Pending Timer (PPT(N))
- Join Expiry Timer (ET(N))

Per downstream (Port, S, G, rpt):

- DownstreamJPRptState: One of { "NoInfo" (NI), "Pruned" (P), "Prune Pending" (PP) }

Per downstream (Port, S, G, rpt, N):

- Prune Pending Timer (PPT(N))
- Join Expiry Timer (ET(N))

Where S is the address of the multicast source, G is the Group address and N is the upstream neighbor field in the Join/Prune message. Notice that unlike on PIM-SM routers where PPT and ET are per (Interface, S, G), PIM Snooping switches have to maintain PPT and ET per (Port, S, G, N). The reasons for this are explained in section 3.3.2.

Apart from the above states, we define the following state summarization macros.

UpstreamNeighbors(*,G): If there is one or more Join(*,G) received on any port with upstream neighbor N and ET(N) is active, then N is added to UpstreamNeighbors(*,G). This set is used to determine if a Join(*,G) or a Prune(*,G) with upstream neighbor N needs to be sent upstream.

UpstreamNeighbors(S,G): If there is one or more Join(S,G) received on any port with upstream neighbor N and ET(N) is active, then N is added to UpstreamNeighbors(S,G). This set is used to determine if a Join(S,G) or a Prune(S,G) with upstream neighbor N needs to be sent upstream.

UpstreamPorts(G): This is the set of all ports on which the Upstream Neighbors in UpstreamNeighbors(*,G) and the UpstreamNeighbors(S,G) for the given G are learnt. This set is used in section 3.3.8. to determine the ports on which Join/Prune messages must be sent. Data traffic MUST also be forwarded to these ports to facilitate assert election in the VPLS.

PWPorts: This is the set of all PWs.

OutgoingPortList(*,G): This is the set of all ports to which traffic needs to be forwarded on a (*,G) match. Split Horizon rules apply as noted in section 3.8.

OutgoingPortList(S,G): This is the set of all ports to which traffic needs to be forwarded on an (S,G) match. Split Horizon rules apply as noted in section 3.8.

NumETsActive(Port,*,G): Number of (Port,*,G,N) entries that have Expiry Timer running. This macro keeps track of the number of Join(*,G)s that are received on this Port with different upstream neighbors.

NumETsActive(Port,S,G): Number of (Port,S,G,N) entries that have Expiry Timer running. This macro keeps track of the number of Join(*,G)s that are received on this Port with different upstream neighbors.

JoinAttributes(*,G): Join attributes [PIM-JOIN-ATTR] are TLVs that may be present in received Join(*,G) messages. If present, they must be copied to JoinAttributes(*,G).

JoinAttributes(S,G): Join attributes [PIM-JOIN-ATTR] are TLVs that may be present in received Join(S,G) messages. If present, they must be copied to JoinAttributes(S,G).

Since there are a few differences between the downstream state machines of PIM-SM Routers and PIM-SM snooping switches, we specify the details of the downstream state machine of PIM-SM snooping switches at the risk of repeating most of the text documented in [PIM-SM].

3.3.2. Explaination for per (S,G,N) states

In PIM Routing protocols, states are built per (S,G). On a router, an (S,G) has only one RPF-Neighbor. However, a PIM Snooping switch does not have the Layer 3 routing information available to the routers in order to determine the RPF-Neighbor for a multicast flow. It merely discovers it by snooping the Join/Prune message. A PE could have snooped on two or more different Join/Prune messages for the same (S,G) that could have carried different Upstream-Neighbor fields. This could happen during transient network conditions or due to dual-homed sources. A PE cannot make assumptions on which one to pick, but instead must facilitate the CE routers decide which Upstream Neighbor gets elected the RPF-Neighbor. And for this purpose, the PE will have to track downstream and upstream Join/Prune states per (S,G,N).

3.3.3. Receiving (*,G) PIM-SM Join/Prune Messages

A Join(*,G) or Prune(*,G) is "received" when the port on which it was received is not also the port on which the upstream-neighbor N of the Join/Prune(*,G) was learnt.

When a router receives a Join(*,G) or a Prune(*,G) with upstream neighbor N, it must process the message as defined in the state machine below. Note that the macro computations of the various macros resulting from this state machine transition is exactly as specified in the PIM-SM RFC [PIM-SM].

We define the following per-port (*,G,N) macro to help with the state machine below.

Figure 1: Downstream per-port (*,G) state machine in tabular form

	Previous State		
Event	NoInfo (NI)	Join (J)	Prune-Pend
Receive Join(*,G)	-> J state Action RxJoin(N)	-> J state Action RxJoin(N)	-> J state Action RxJoin(N)
Receive Prune(*,G) and NumETsActive<=1	-	-> PP state Start PPT(N)	-> PP state
Receive Prune(*,G) and NumETsActive>1	-	-> J state Start PPT(N)	-
PPT(N) expires	-	-> J state Action PPTExpiry(N)	-> NI state Action PPTExpiry(N)
ET(N) expires and NumETsActive<=1	-	-> NI state Action ETExpiry(N)	-> NI state Action ETExpiry(N)
ET(N) expires and NumETsActive>1	-	-> J state Action ETExpiry(N)	-> NI state Action ETExpiry(N)

Action RxJoin(N):

If ET(N) is not already running, then start ET(N). Otherwise restart ET(N).
If N is not already in UpstreamNeighbors(*,G), then add N to UpstreamNeighbors(*,G) and trigger a Join(*,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8.
Record N as RPF_Neighbor(*,G).
If there are Join Attributes in the received (S,G) message and if the Join Attributes are different from the recorded JoinAttributes(S,G), then copy them into JoinAttributes(S,G). Also trigger a Join(S,G)

with upstream neighbor N to be forwarded upstream as specified in section 3.3.8.

Action PPTExpiry(N):

Disable timers ET(N) and PPT(N). If there are no other (Port,*,G) states with NumETsActive(Port,*,G) > 0, then trigger a Prune(*,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8. Then delete N from UpstreamNeighbors(*,G).

Send a Prune-Echo(*,G) with upstream-neighbor N on the downstream port.

Action ETExpiry(N):

Disable timers ET(N) and PPT(N). If there are no other (Port,*,G) states with NumETsActive(Port,*,G) > 0, then trigger a Prune(*,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8. Then delete N from UpstreamNeighbors(*,G).

3.3.4. Receiving (S,G) PIM-SM Join/Prune Messages

A Join(S,G) or Prune(S,G) is "received" when the port on which it was received is not also the port on which the upstream-neighbor N of the Join/Prune(S,G) was learnt.

When a router receives a Join(S,G) or a Prune(S,G) with upstream neighbor N, it must process the message as defined in the state machine below. Note that the macro computations of the various macros resulting from this state machine transition is exactly as specified in the PIM-SM RFC [PIM-SM].

Figure 2: Downstream per-port (S,G) state machine in tabular form

	Previous State		
Event	NoInfo (NI)	Join (J)	Prune-Pend
Receive Join(S,G)	-> J state Action RxJoin(N)	-> J state Action RxJoin(N)	-> J state Action RxJoin(N)
Receive Prune (S,G) and NumETsActive<=1	-	-> PP state Start PPT(N)	-> PP state
Receive Prune(S,G) and NumETsActive>1	-	-> J state Start PPT(N)	-
PPT(N) expires	-	-> J state Action PPTExpiry(N)	-> NI state Action PPTExpiry(N)
ET(N) expires and NumETsActive<=1	-	-> NI state Action ETExpiry(N)	-> NI state Action ETExpiry(N)
ET(N) expires and NumETsActive>1	-	-> J state Action ETExpiry(N)	-> NI state Action ETExpiry(N)

Action RxJoin(N):

If ET(N) is not already running, then start ET(N). Otherwise, restart

ET(N).

If N is not already in UpstreamNeighbors(S,G), then add N to UpstreamNeighbors(S,G) and trigger a Join(S,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8.
Record N as RPF_Neighbor(S,G).
If there are Join Attributes in the received (S,G) message and if the Join Attributes are different from the recorded JoinAttributes(S,G), then copy them into JoinAttributes(S,G). Also trigger a Join(S,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8.

Action PPTExpiry(N):

Disable timers ET(N) and PPT(N). If there are no other (Port,S,G) states with NumETsActive(Port,S,G) > 0, then trigger a Prune(S,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8. Then delete N from UpstreamNeighbors(S,G).

Send a Prune-Echo(S,G) with upstream-neighbor N on the downstream port.

Action ETExpiry(N):

Disable timers ET(N) and PPT(N). If there are no other (Port,S,G) states with NumETsActive(Port,S,G) > 0, then trigger a Prune(S,G) with upstream neighbor N to be forwarded upstream as specified in section 3.3.8. Then delete N from UpstreamNeighbors(S,G).

3.3.5. Receiving (S,G,rpt) Join/Prune Messages

A Join(S,G,rpt) or Prune(S,G,rpt) is "received" when the port on which it was received is not also the port on which the upstream-neighbor N of the Join/Prune(S,G,rpt) was learnt.

While it is important to ensure that the (S,G) and (*,G) state machines allow for handling per (S,G,N) states, it is not as important for (S,G,rpt) states. It suffices to say that the downstream (S,G,rpt) state machine is the same as what is defined in section 4.5.4 of the PIM-SM RFC [PIM-SM].

3.3.6. Sending (*,G) Join/Prune Messages

A PIM Snooping PE MUST implement the Upstream (*,G) state machine for which the procedures are similar to what is defined in section 4.5.6 of [PIM-SM]. Section 3.3.8. of this draft specifies how the message should be sent.

For the purposes of the Upstream (*,G) state machine, a Join(*,G) or Prune(*,G) message with upstream neighbor N is "seen" on a PIM Snooping switch if the port on which the message was received is also the port on which the upstream neighbor N was learnt.

3.3.7. Sending (S,G) Join/Prune Messages

A PIM Snooping PE MUST implement the Upstream (S,G) state machine for which the procedures are similar to what is defined in section 4.5.6 of [PIM-SM]. Section 3.3.8. of this draft specifies how the message should be sent.

For the purposes of the Upstream (S,G) state machine, a Join(*,G) or Prune(*,G) message with upstream neighbor N is "seen" on a PIM Snooping switch if the port on which the message was received is also the port on which the upstream neighbor N was learnt.

3.3.8. Sending PIM Join/Prune message upstream.

The downstream Join/Prune state machines above describe when PIM Join/Prune packets must be forwarded upstream and with what upstream neighbor field. In order to correctly facilitate assert among the CE routers, such Join/Prunes need to sent not only towards the upstream neighbor, but also on certain PWs.

If JoinAttributes(*,G) is not empty, then it must be encoded in a Join(*,G) message sent upstream.

If JoinAttributes(S,G) is not empty, then it must be encoded in a Join(S,G) message sent upstream.

If the Join/Prune message being sent out is a refresh Join(*,G) message, then send the refresh Join(*,G) on all ports in UpstreamPorts(G). The Upstream Neighbor field should be the recorded RPF_Neighbor(*,G).

If the Join/Prune message being sent out is a refresh Join(S,G) message, then send the refresh Join(S,G) on all ports in UpstreamPorts(G). The Upstream Neighbor field should be the recorded RPF_Neighbor(S,G).

If the Join/Prune message being sent out is a triggered Join/Prune message (due to an event in the downstream Join/Prune state machine), then the following rules apply. These rules apply to both (S,G) and (*,G) Join/Prune messages to be sent out:

- The upstream neighbor field N in the Join/Prune to be sent is dictated by the downstream Join/Prune state machine transition.
- If the downstream Join/Prune event was on an AC port, then send the upstream Join/Prune message to all PWs in UpstreamPorts(G). Send the Join/Prune message to Rport(N) also.
- If the downstream Join/Prune event was on a PW port and if Rport(N) is a PW, then silently discard the Join/Prune

message without sending it. If Rport(N) is an AC, then send the Join/Prune message on that AC.

The source IP address in PIM packets sent upstream SHOULD be the address of a PIM neighbor in the VPLS. The address picked MUST NOT be the upstream neighbor field to be encoded in the packet. The layer 2 encapsulation for the selected source IP address MUST be the encapsulation recorded in the PIM Neighbor database for that IP address.

3.3.9. Triggering ASSERT Election in PIM-SM

In PIM-SM, there are scenarios where multiple routers could be forwarding the same multicast traffic on a LAN. When this happens, using PIM Assert Election process by sending PIM Assert Messages, routers ensure that only the Assert Winner forwards traffic on the LAN. In a typical LAN, the Assert Election is a data driven event and happens only if a router sees traffic on the interface to which it should be forwarding the traffic. Therefore, in the case of VPLS, in order to trigger Assert Election and stop duplicate traffic, it is necessary that two routers that are forwarding duplicate traffic for an (S,G)/(*,G) see each other's traffic.

PIM Snooping switches must hence ensure that they not only forward multicast traffic for an (S,G) on the ports on which they snooped Joins(S,G)/Joins(*,G), but also on the ports on which such Joins were forwarded (i.e. towards the upstream neighbor(s)). Traffic should not be forwarded on the port on which it was received. So if two or more Joins(S,G) each carrying a different upstream neighbor field were snooped at a PE, then the ports on which such Joins were snooped along with the ports on which the upstream neighbors were learnt must be added to the outgoing port list.

VPLS Split Horizon Rules dictate that traffic arriving on a PW MUST NOT be forwarded onto other PW(s). Let us consider the example in Figure 3: So at a downstream PE (say PE1), if a Join(S,G) with Upstream Neighbor N1 was sent on one PW (say PW12) and another Join(S,G) with Upstream Neighbor N2 was sent on another PW (say PW13), then duplicate traffic will arrive on both PW12 and PW13. Due to VPLS Split Horizon Rules, traffic from PW12 cannot be forwarded onto PW13 and vice-versa. However, as long as the upstream PEs (PE2 and PE3) also snoop on the Joins/Prunes, then per the rules in the previous paragraph, they will add the PW towards both upstream neighbors N1 and N2 to the outgoing port list.

Let us consider the scenario in Figure 3.

Figure 3: An Example Scenario for Triggering Assert

```
                                    +------+ AC3 +------+
                                    |  PE2 |-----| CE3  |
                                   /|      |     |      |
                                  / +------+     +------+
                                 /     |             |
                                /      |             |
                               /PW12   |             |
                              /        |          +-----+
                             /         |PW23      |  S  |
                            /          |          +-----+
                           /           |             |
                          /            |             |
                         /             |             |
  +------+     +------+ /           +------+     +------+
  | CE1  |     |  PE1 |/   PW13     |  PE3 |     | CE4  |
  |      |-----|      |-------------|      |-----|      |
  +------+ AC1 +------+             +------+ AC4 +------+
                   |
                   |AC2
               +------+
               | CE2  |
               |      |
               +------+
```

In the scenario depicted in Figure 3, both CE1 and CE2 have two ECMP routes to reach the source "S". Hence, CE1 may pick CE3 as its next hop ("Upstream Neighbor"), and CE2 may pick CE4 as its next hop. As a result, both CE1 and CE2 will receive duplicate traffic for a moment. If Assert procedures are not invoked by CE3 and CE4, the duplicate traffic on the LAN will persist forever. Following is the sequence of events that illustrates how duplicate traffic is resolved. Note that this illustration assumes PIM Snooping in the VPLS with Join Suppression disabled on the CEs. Procedures for PIM Proxy are slightly different and will be covered in section 3.6.

1. CE1 sends a Join(S,G) with N=CE3.
2. When PE1 snoops on the Join(S,G), it adds CE3 to its UpstreamNeighbors(S,G). It also adds AC1 and PW12 to its OutgoingPortList(S,G). UpstreamNeighbors(S,G) on PE1 = {CE3}. OutgoingPortList(S,G) on PE1 = {AC1, PW12}.
3. PE1 floods the Join(S,G) in the VPLS. If using LDP (as

explained in [VPLS-MCAST-LDP]), PE1 also sends the Join(S,G) via LDP on PW12 (the PW towards UpstreamNeighbors(S,G)).

4. When PE2 receives the Join(S,G), it adds CE3 to its UpstreamNeighbors(S,G) and it also adds PW12 and AC3 to its OutgoingPortList(S,G). UpstreamNeighbors(S,G) on PE2 = {CE3}. OutgoingPortList(S,G) on PE2 = {PW12, AC3}.

The above is all that needs to occur in most cases where there is no assert.

5. CE2 sends a Join(S,G) with N=CE4.
6. This results in PE1 adding CE4 to its UpstreamNeighbors(S,G). It also adds AC2 and PW13 to its OutgoingPortList(S,G). UpstreamNeighbors(S,G) at PE1 = {CE3, CE4}. OutgoingPortList(S,G) on PE2 = {AC1, AC2, PW12, PW13}.
7. PE1 floods the join in the VPLS. If using LDP, since the Join was received on an AC AND since a neighbor was added to UpstreamNeighbors(S,G), it sends a Join(S,G) via LDP on the PWs towards UpstreamNeighbors(S,G) {PW12, PW13}.
8. PE2 receives the Join(S,G) and adds CE4 to UpstreamNeighbors(S,G). It also adds PW23 to its OutgoingPortList(S,G). UpstreamNeighbors(S,G) = {CE3, CE4}. OutgoingPortList(S,G) on PE2 = {PW12, AC3, PW23}.
9. PE3 receives the Join(S,G) too. It adds CE4 to its UpstreamNeighbors(S,G). And it adds PW13 and AC4 to its OutgoingPortList(S,G). UpstreamNeighbors(S,G) = {CE4}. OutgoingPortList(S,G) on PE3 = {PW13, AC4}.

So even before duplicate traffic starts flowing, the Outgoing interface list on the PEs are (i.e., the forwarding plane):

PE1: {AC1, AC2, PW12, PW13}
PE2: {PW12, AC3, PW23}
PE3: {PW13, AC4}

By building such a forwarding state when Joins are processed, there needs to be no additional action taken by the PEs when duplicate traffic is received. Traffic arriving from CE3 will be forwarded to CE4 thus allowing for CE3 and CE4 to transparently trigger assert election. This makes the PIM Snooping completely a control-plane protocol without any data-plane interaction. When the assert election is complete, if CE3 becomes the assert winner, then

10. CE2 sends a Prune(S,G) with N=CE4 and a Join(S,G) with N=CE3.
11. When PE1 snoops the Prune(S,G), it removes CE4 from its UpstreamNeighbors(S,G). It also removes AC2 and PW13 from the OutgoingPortList(S,G). The Join(S,G) will result in AC2 added back to OutgoingPortList(S,G). OutgoingPortList(S,G) on PE1 = {AC1, AC2, PW12}.

12. PE1 floods the messages in the VPLS. If LDP is used, since the Prune was received on an AC and since a neighbor was removed from UpstreamNeighbors(S,G), the Prune(S,G)

will be sent on the PWs towards UpstreamNeighbors(S,G) {PW12, PW13}. The Join(S,G) need not be sent to {PW12} since the CE3 was already in UpstreamNeighbors(S,G).

13. PE2 receives the Prune directed to CE4. As a result, PE2 removes CE4 from its UpstreamNeighbors(S,G). It also removes PW23 from its OutgoingPortList(S,G).
UpstreamNeighbors(S,G) on PE2 = {CE3}.
OutgoingPortList(S,G) on PE2 = {PW12, AC3}.
14. PE3 receives the Prune too. As a result, PE3 removes CE4 from its UpstreamNeighbors(S,G). It also removes PW13 and AC4 from its OutgoingPortList(S,G). So, PE3 purges state for that (S,G).

After assert election, the forwarding state should be:

PE1: {AC1, PW12}
PE2: {PW12, AC2}
PE3: {}

Other more complex duplicate traffic scenarios can exist due to the existence of (S,G) and/or (*,G) states and/or IGMP receiver states in a VPLS. The inheritance rules in PIM-SM and the rules specified in this draft should ensure that assert is triggered among the CEs in all scenarios.

3.4. Bidirectional-PIM (PIM-BIDIR)

PIM-BIDIR is a variation of PIM-SM. The main differences between PIM-SM and Bidirectional-PIM are as follows:

- There are no source-based trees, and source-specific multicast is not supported (i.e., no (S,G) states) in PIM-BIDIR.
- Multicast traffic can flow up the shared tree in PIM-BIDIR.
- To avoid forwarding loops, one router on each link is elected as the Designated Forwarder (DF) for each RP in PIM-BIDIR.

The main advantage of PIM-BIDIR is that it scales well for many-to-many applications. However, the lack of source-based trees means that multicast traffic is forced to remain on the shared tree.

The procedures to discover PIM-SM routers in a VPLS instance are as described in section 3.2. For PIM-BIDIR to work properly, all routers within the domain must know the address of the RP. During RP discovery time, PIM routers elect DF per subnet for each RP. The algorithm to elect the DF is as follows: all PIM neighbors in a subnet advertise their unicast route to elect the RP and the router with the best route is elected.

Snooping for PIM-BIDIR is much simpler than it is for PIM-SM. The complexity resulting from various combinations of (S,G), (*,G), IGMP and assert states makes PIM-SM procedures fairly complex. PIM-BIDIR has none of those issues since PIM-BIDIR builds only (*,G) states and all routers on a LAN agree on who the upstream neighbor, i.e. DF(RP) is. So the snooping procedures for PIM-BIDIR is very much like that on a PIM-BIDIR router [PIM-BIDIR].

3.4.1. Building PIM-BIDIR Snooping States

The PEs MUST snoop on PIM-BIDIR Join/Prune messages and build states as described in [PIM-BIDIR]. The PEs SHOULD simply flood all other PIM packet types. Since there are no special procedures required for PIM-BIDIR snooping, we simply refer the reader to the [PIM-BIDIR] draft.

3.5. PIM-DM

The characteristics of PIM-DM is flood and prune behavior. Shortest path trees are built as a multicast source starts transmitting.

The procedures to discover PIM-DM routers are as explained in section 3.2.

3.5.1. Building PIM-DM Snooping States

PIM-DM Snooping states are built by snooping on the PIM-DM Join, Prune, Graft and State Refresh messages received on AC/PWs and State-Refresh Messages sent on AC/PWs. By snooping on these PIM-DM messages, a PE builds the following states per (S,G,N) where S is the address of the multicast source, G is the Group address and N is the upstream neighbor to which Prunes/Grafts are sent by downstream CEs:

Per PIM (S,G,N):

Per Port PIM (S,G,N) Prune State:

- DownstreamPState(S,G,N,Port): One of {"NoInfo" (NI), "Pruned" (P), "PrunePending" (PP)}
- Prune Pending Timer (PPT)
- Prune Timer (PT)
- Upstream Port (valid if the PIM(S,G,N) Prune State is "Pruned").

3.5.2.   PIM-DM Downstream Per-Port PIM(S,G,N) State Machine

The downstream per-port PIM(S,G,N) state machine is as defined in section 4.4.2 of [PIM-DM] with a few changes relevant to PIM Snooping. When reading section 4.4.2 of [PIM-DM] for the purposes of PIM-Snooping please be aware that the downstream states are built per

(S, G, N, Downstream-Port} in PIM-Snooping and not per {Downstream-Interface, S, G} as in a PIM-DM router. As noted in the previous section 3.5.1. , the states (DownstreamPState) and timers (PPT and PT) are per (S,G,N,P).

3.5.3. Triggering ASSERT election in PIM-DM

Since PIM-DM is a flood-and-prune protocol, traffic is flooded to all routers unless explicitly pruned. Since PIM-DM routers do not prune on non-RPF interfaces, PEs should typically not receive Prunes on Rport(RPF-neighbor). So the asserting routers should typically be in pim_oiflist(S,G). In most cases, assert election should occur naturally without any special handling since data traffic will be forwarded to the asserting routers.

However, there are some scenarios where a prune might be received on a port which is also an upstream port (UP). If we prune the port from pim_oiflist(S,G), then it would not be possible for the asserting routers to determine if traffic arrived on their downstream port. This can be fixed by adding pim_iifs(S,G) to pim_oiflist(S,G) so that data traffic flows to the UP ports.

3.6. PIM Proxy

As noted earlier in section 2.4. , PIM Snooping will work correctly only if Join Suppression is disabled in the VPLS. If Join Suppression is enabled in the VPLS, then PEs MUST do PIM Proxy for VPLS Multicast to work correctly.

A PIM Proxy switch behaves like a PIM Router by doing most of the functionality of a PIM Router. The complexity however is much lesser on a switch since many of the issues that a PIM Router has to deal with are not relevant on a switch. A PIM Router needs to be able to build and maintain RP-Sets. They also have to deal with the Register and Assert State Machines. There are other complexities for a PIM Router resulting from inter-domain multicast. A PIM Snooping or PIM Proxy switch can be agnostic of all of this. All that a PIM Proxy switch cares about is building multicast states using PIM Hellos and PIM Join/Prune message. As such it's complexity is greatly reduced.

Other than the procedures defined here, the rest of the procedures that apply to PIM Snooping apply to PIM Proxy as well.

3.6.1. Downstream PIM Proxy behavior

A PIM-SM or PIM-BIDIR Proxy PE is interested in the Hello and Join/Prune messages. The proposed PIM Proxy solution for PIM-SM and PIM-BIDIR is to proxy only Join/Prune messages. PIM Proxy for PIM-DM is for future study.

PIM Hellos MUST be snooped while being flooded in the VPLS.

PIM Join/Prune messages arriving at an AC MUST be consumed. If [VPLS-MCAST-LDP] is not used to distribute multicast states among the PEs, then PIM Join/Prune messages arriving at a PW MUST also be consumed.

All other PIM packet types are flooded in the VPLS without needing observation.

Performing only proxy of Join/Prune messages keeps the switch behavior very similar to that of a PIM router without introducing too much additional complexity. It keeps the PIM Proxy solution fairly simple. Since Join/Prunes are forwarded by a PE along the slow-path and all other PIM packet types are forwarded along the fast-path, it is very likely that packets forwarded along the fast-path will arrive "ahead" of Join/Prune packets at a CE router (note the stress on the fact that fast-path messages will never arrive after Join/Prunes). Of particular importance are Hello packets sent along the fast-path. We can construct a variety of scenarios resulting in out of order delivery of Hellos and Join/Prune messages. However, there should be no deviation from normal expected behavior observed at the CE router receiving these messages out of order.

The other option for a PIM Proxy solution is to proxy both Hello and Join/Prune messages that a PE is interested in building states for. If Hellos are being proxied, then it becomes necessary that the PE proxy all other PIM packet types also. Because if Hellos are received after other packet types are received at a CE router, then bad things will happen. That means every PIM packet has to be sent along the slow-path. This greatly increases the complexity on the CE router, it is very compute intensive and does not scale well. Also, proxying Hellos will result in added latency to delivery of Hello messages to a CE and that affects multicast convergence in the VPLS.

3.6.2. Upstream PIM Proxy behavior

Since a PIM Proxy switch consumes Join/Prune messages, it must also originate PIM Join/Prune messages to be sent upstream. If [VPLS-MCAST-LDP] is employed, then triggered Join/Prune messages are sent via LDP to forward PIM Join/Prunes on PWs. Join/Prune messages need not be refreshed on PWs when [VPLS-MCAST-LDP] is employed. On ACs, both triggered and refresh Join/Prunes are forwarded as PIM packets.

The source IP address in PIM packets sent upstream SHOULD be the address of a PIM neighbor in the VPLS. The address picked MUST NOT be the upstream neighbor field to be encoded in the packet. The layer 2 encapsulation for the selected source IP address MUST be the encapsulation recorded in the PIM Neighbor database for that IP address.

If Explicit Tracking is disabled in the VPLS, then it does not matter what Source IP Address is picked in the packets sent upstream as long as we adhere to the rule in the previous paragraph. However, if Explicit Tracking is enabled in the VPLS, then we cannot do this. In fact, it would be wrong to even enable PIM Proxy in the VPLS since the CE routers may wish to receive the Join/Prunes from every interested CE router. If a PE determines that Explicit Tracking is enabled in the VPLS, it MUST disable PIM Proxy and SHOULD follow PIM Snooping procedures instead.

3.7. Directly Connected Multicast Source

If there is a source in the CE network that connects directly into the VPLS instance, then multicast traffic from that source MUST be sent to all PIM routers on the VPLS instance apart from the igmp receivers in the VPLS. If there is already (S,G) or (*,G) snooping state that is formed on any PE, this will not happen per the current forwarding rules and guidelines. So, in order to determine if traffic needs to be flooded to all routers, a PE must be able to determine if the traffic came from a host on that LAN. There are three ways to address this problem:

- The PE would have to do ARP snooping to determine if a source is directly connected.
- Another option is to have configuration on all PEs to say there are CE sources that are directly connected to the VPLS instance and disallow snooping for the groups for which the source is going to send traffic. This way traffic from that source to those groups will always be flooded within the provider network.
- A third option is to require that sources of CE multicast routers must appear behind a router.

3.8. Data Forwarding Rules

First we define the rules that are common to PIM-SM, PIM-BIDIR and PIM-DM PEs. Forwarding rules for each protocol type is specified in the sub-sections.

If there is no matching forwarding state, then the PE MAY either discard the packet or send it towards all the snooped PIM CE routers or to a configured set of ports. How this is determined is outside the scope of this document.

The following rules MUST be followed when forwarding multicast traffic in a VPLS:

- Traffic arriving on a port MUST NOT be forwarded back onto the same port.

- Due to VPLS Split-Horizon rules, traffic ingressing on a PW MUST NOT be forwarded to any other PW.

3.8.1. PIM-SM Data Forwarding Rules

Per the rules in [PIM-SM] and per the additional rules specified in this document,

```
OutgoingPortList(*,G) = inherited_olist(*,G) (+) UpstreamPorts(G)
                       (+) Rport(PimDR)

OutgoingPortList(S,G) = inherited_olist(S,G) (+) UpstreamPorts(G)
                        (+) Rport(PimDR)
```

[PIM-SM] specifies how inherited_olist(*,G) and inherited_olist(S,G) are built. PimDR is the IP address of the PIM DR in the VPLS.

The PIM-SM Snooping forwarding rules are defined below in pseudocode:

```
BEGIN
   iif is the incoming port of the multicast packet.
   S is the Source IP Address of the multicast packet.
   G is the Destination IP Address of the multicast packet.

   If there is (S,G) state on the PE
   Then
       OutgoingPortList = OutgoingPortList(S,G)
   Else if there is (*,G) state on the PE
   Then
       OutgoingPortList = OutgoingPortList(*,G)
   Else
       OutgoingPortList = UserDefinedPortList
   Endif

   If iif is an AC
   Then
       OutgoingPortList = OutgoingPortList (-) iif
   Else
       ## iif is a PW
       OutgoingPortList = OutgoingPortList (-) PWPorts
   Endif

   Forward the packet to OutgoingPortList.
END
```

First if there is (S,G) state on the PE, then the set of outgoing ports is OutgoingPortList(S,G).

Otherwise if there is (*,G) state on the PE, the set of outgoing ports is OutgoingPortList(*,G).

The packet is forwarded to the selected set of outgoing ports while observing the rules above in section 3.8.

3.8.2. PIM-BIDIR Data Forwarding Rules

The PIM-BIDIR Snooping forwarding rules are defined below in pseudocode:

```
BEGIN
   iif is the incoming port of the multicast packet.
   G is the Destination IP Address of the multicast packet.

   If there is forwarding state for G
   Then
       OutgoingPortList = olist(G)
   Else
       OutgoingPortList = UserDefinedPortList
   Endif

   If iif is an AC
   Then
       OutgoingPortList = OutgoingPortList (-) iif
   Else
       ## iif is a PW
       OutgoingPortList = OutgoingPortList (-) PWPorts
   Endif

   Forward the packet to OutgoingPortList.
END
```

If there is forwarding state for G, then forward the packet to olist(G) while observing the rules above in section 3.8.

[PIM-BIDIR] specifies how olist(G) is contructed.

3.8.3. PIM-DM Data Forwarding Rules

The PIM-DM Snooping data forwarding rules are defined below in pseudocode:

```
BEGIN
   iif is the incoming port of the multicast packet.
   S is the Source IP Address of the multicast packet.
   G is the Destination IP Address of the multicast packet.

   If there is (S,G) state on the PE
   Then
       OutgoingPortList = olist(S,G)
   Else
```

```
OutgoingPortList = UserDefinedPortList
```

```
    Endif

    If iif is an AC
    Then
        OutgoingPortList = OutgoingPortList (-) iif
    Else
        ## iif is a PW
        OutgoingPortList = OutgoingPortList (-) PWPorts
    Endif

    Forward the packet to OutgoingPortList.
END
```

If there is forwarding state for (S,G), then forward the packet to olist(S,G) while observing the rules above in section 3.8.

[PIM-DM] specifies how olist(S,G) is contructed.

4. IANA Considerations

This document does not require any IANA assignments or action.

5. Security Considerations

Security considerations provided in VPLS solution documents (i.e., [VPLS-LDP] and [VPLS-BGP) apply to this document as well.

6. References

6.1. Normative References

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

[PIM-DM]   Deering, S., et al. "Protocol Independent Multicast Version 2 - Dense Mode Specification", RFC 3973, January 2005.

[PIM-SM]   Fenner, W, et al. "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (Revised)", draft-ietf-pim-sm-v2-new-11.txt, April 2005.

[PIM-SSM]   Holbrook, H., et al. "Source-Specific Multicast for IP", work in progress

[PIM-BIDIR]   Handley, M., et al. "Bi-directional Protocol Independent Multicast (BIDIR-PIM)", work in progress

[PIM-JOIN-ATTR] Boers, A, et al, "Format for using TLVs in PIM messages", draft-ietf-pim-join-attributes-01.txt, Work in progress

6.2. Informative References

[VPLS-LDP]   Lasserre, M, et al. "Virtual Private LAN Services over MPLS", work in progress

[VPLS-BGP]   Kompella, K, et al. "Virtual Private LAN Service", work in progress

[IGMP-SNOOP]   Christensen, M., et al. "Considerations for IGMP and MLD Snooping Switches", work in progress

[VPLS-MCAST-LDP] Qui, R, Serbest, Y, et al, "Using LDP for VPLS Multicast", draft-qiu-serbest-vpls-mcast-ldp-00.txt, Work in progress

[VPLS-MCAST-BGP] Aggarwal, R, et al, "Propagation of VPLS IP Multicast Group Membership Information", draft-raggarwa-l2vpn-vpls-mcast-ctrl-00.txt, Work in progress

[VPLS-MCAST-TREES] Aggarwal, R, et al. "Multicast in VPLS", draft-raggarwa-l2vpn-vpls-mcast-01.txt, Work in progress.

Authors' Addresses

Venu Hemige
Alcatel North America
701 East Middlefield Rd.
Mountain View, CA 94043
Venu.hemige@alcatel.com

Yetik Serbest
AT&T Labs
9505 Arboretum Blvd.
Austin, TX 78759
Yetik_serbest@labs.sbc.com

Ray Qiu
Alcatel North America
701 East Middlefield Rd.
Mountain View, CA 94043
Ray.Qiu@alcatel.com

Suresh Boddapati
Alcatel North America
701 East Middlefield Rd.
Mountain View, CA 94043
Suresh.boddapati@alcatel.com

Rob Nath
Riverstone Networks
5200 Great America Parkway

Santa Clara, CA 95054
Rnath@riverstonenet.com

Sunil Khandekar
Alcatel North America
701 East Middlefield Rd.
Mountain View, CA 94043
Sunil.khandekar@alcatel.com

Vach Kompella
Alcatel North America
701 East Middlefield Rd.
Mountain View, CA 94043
Vach.kompella@alcatel.com

Marc Lasserre
Riverstone Networks
Marc@riverstonenet.com

Himanshu Shah
Ciena
hshah@ciena.com

Intellectual Property Statement

Full copyright statement