Network Working Group
Internet Draft
Expiration Date: June 2005

R. Aggarwal (Juniper)
D. Papadimitriou (Alcatel)
S. Yasukawa (NTT)
Editors

Extensions to RSVP-TE for Point to Multipoint TE LSPs

draft-ietf-mpls-rsvp-te-p2mp-01.txt

Status of this Memo

Abstract

This document describes extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for the setup of Traffic Engineered (TE) point-to-multipoint (P2MP) Label Switched Paths (LSPs) in Multi-Protocol Label Switching (MPLS) and Generalized MPLS (GMPLS) networks. The solution relies on RSVP-TE without requiring a multicast routing protocol in the Service Provider core. Protocol elements and procedures for this solution are described. There can be various applications for P2MP TE LSPs such as IP multicast. Specification of how such applications will use a P2MP TE LSP is outside the scope of this document.

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 [KEYWORDS].

Authors' Note

Some of the text in the document needs further discussion between authors and feedback from MPLS WG. This has been pointed out when applicable. A change log and reviewed/updated text will be made available online.

Table of Contents

1. Terminology

This document uses terminologies defined in [RFC3031], [RFC2205], [RFC3209], [RFC3473] and [P2MP-REQ]. In particular, this document uses the notation defined in [P2MP-REQ] for describing the components on a P2MP LSP between root, branches and leaves.

2. Introduction

[RFC3209] defines a mechanism for setting up point-to-point (P2P) Traffic Engineered (TE) LSPs in MPLS networks. [RFC3473] defines extensions to [RFC3209] for setting up P2P TE LSPs in GMPLS networks. However these specifications do not provide a mechanism for building point-to-multipoint P2MP TE LSPs.

This document defines extensions to RSVP-TE [RFC3209] and [RFC3473] protocol to support P2MP TE LSPs satisfying the set of requirements described in [P2MP-REQ].

This document relies on the semantics of RSVP that RSVP-TE inherits for building P2MP LSP Tunnels. A P2MP LSP Tunnel is comprised of multiple S2L sub-LSPs. These S2L sub-LSPs are set up between the ingress and egress LSRs and are appropriately combined by the branch LSRs using RSVP semantics to result in a P2MP TE LSP. One Path message may signal one or multiple S2L sub-LSPs. Hence the S2L sub-LSPs belonging to a P2MP LSP Tunnel can be signaled using one Path message or split across multiple Path messages.

Path computation and P2MP application specific aspects are outside of the scope of this document.

3. Mechanism

This document describes a solution that optimizes data replication by allowing non-ingress nodes in the network to be replication/branch nodes. A branch node is a LSR that is capable of replicating the incoming data on two or more outgoing interfaces. The solution uses RSVP-TE in the core of the network for setting up a P2MP TE LSP.

The P2MP TE LSP is set up by associating multiple S2L TE sub-LSPs and relying on data replication at branch nodes. This is described further in the following sub-sections by describing P2MP tunnels and how they relate to S2L sub-LSPs.

3.1. P2MP Tunnels

The specific aspect related to P2MP TE LSP is the action required at a branch node, where data replication occurs. Incoming labeled data is appropriately replicated to several outgoing interfaces which may have different labels.

A P2MP TE tunnel comprises of one or more P2MP LSPs referred to as P2MP LSP tunnels. A P2MP TE Tunnel is identified by a P2MP SESSION object. This object contains an identifier of the P2MP session defined as a P2MP ID, a tunnel ID and an extended tunnel ID.

The fields of a P2MP SESSION object are identical to those of the SESSION object defined in [RFC3209] except that the Tunnel Endpoint Address field is replaced by the P2MP Identifier (P2MP ID) field. The P2MP ID provides an identifier for the set of destinations of the P2MP TE Tunnel. The P2MP SESSION object is defined in section 20.1.

3.2. P2MP LSP Tunnel

A P2MP LSP Tunnel is identified by the combination of the P2MP ID, Tunnel ID, and Extended Tunnel ID that are part of the P2MP SESSION object, and the tunnel sender address and LSP ID fields of the P2MP SENDER_TEMPLATE object. The new P2MP SENDER_TEMPLATE object is defined in section 20.2.

3.3. Sub-Groups

As with all other RSVP controlled LSP Tunnels, P2MP LSP Tunnel state is managed using RSVP messages. While use of RSVP messages is the same, P2MP LSP Tunnel state differs from P2P LSP state in a number of ways. A notable difference is that a P2MP LSP Tunnel is comprised of multiple S2L Sub-LSPs As a result of this, it may not be possible to signal a P2MP LSP Tunnel in a single RSVP-TE Path/Resv message. It is also possible that such a signaling message can not fit into a single IP packet. It must also be possible to efficiently add and remove endpoints to and from P2MP TE LSPs. An additional issue is that P2MP LSP Tunnels must also handle the state "remerge" problem [P2MP-REQ].

These differences in P2MP state are addressed through the addition of a sub-group identifier (Sub-Group ID) and sub-group originator (Sub-Group Originator ID) to the SENDER_TEMPLATE and FILTER_SPEC objects. Taken together the Sub-Group ID and Sub-Group Originator ID are referred to as the Sub-Group fields.

The Sub-Group fields, together with rest of the SENDER_TEMPLATE and SESSION objects, are used to represent a portion of a P2MP LSP Tunnel's state. The portion of P2MP LSP Tunnel state identified by

specific subgroup field values is referred to as a signaling sub-tree. It is important to note that the term "signaling sub-tree" refers only to signaling state and not data plane replication or branching. For example, it is possible for a node to "split" signaling state for a P2MP LSP Tunnel, but to not branch the data associated with the P2MP LSP Tunnel. Typical applications for generation and use of multiple subgroups are adding an egress and semantic fragmentation to ensure that a Path message remains within a single IP packet.

3.4. S2L Sub-LSPs

A P2MP LSP Tunnel is constituted of one or more S2L sub-LSPs.

3.4.1. Representation of a S2L Sub-LSP

A S2L sub-LSP exists within the context of a P2MP LSP Tunnel. Thus it is identified by the P2MP ID, Tunnel ID, and Extended Tunnel ID that are part of the P2MP SESSION, the tunnel sender address and LSP ID fields of the P2MP SENDER_TEMPLATE object, and the S2L sub-LSP destination address that is part of the S2L_SUB_LSP object. The S2L_SUB_LSP object is defined in section 20.3.

Additionally, a sub-LSP ID contained in the S2L_SUB_LSP object may be used depending on further discussions about the make-before-break procedures described in section 14.

An EXPLICIT_ROUTE Object (ERO) or SUB_EXPLICIT_ROUTE Object (SERO) is used to optionally specify the explicit route of a S2L sub-LSP. Each ERO or a SERO that is signaled corresponds to a particular S2L_SUB_LSP object. Details of explicit route encoding are specified in section 3.5.

3.4.2. S2L Sub-LSPs and Path Messages

The mechanism in this document allows a P2MP LSP Tunnel to be signaled using one or more Path messages. Each Path message may signal one or more S2L sub-LSPs. Support for multiple Path messages is desirable as one Path message may not be large enough to fit all the S2L sub-LSPs; and they also allow separate manipulation of sub-trees of the P2MP LSP Tunnel. The reason for allowing a single Path message, to signal multiple S2L sub-LSPs, is to optimize the number of control messages needed to setup a P2MP LSP Tunnel.

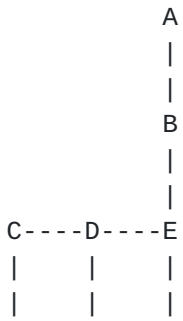
3.5. Explicit Routing

When a Path message signals a single S2L sub-LSP (that is, the Path message is only targeting a single leaf in the P2MP tree), the EXPLICIT_ROUTE object may encode the path to the egress LSR. The Path message also includes the S2L_SUB_LSP object for the S2L sub-LSP being signaled. The < [<EXPLICIT_ROUTE>], <S2L_SUB_LSP> > tuple represents the S2L sub-LSP. The absence of the ERO should be interpreted as requiring hop-by-hop routing for the sub-LSP based on the S2L sub-LSP destination address field of the S2L_SUB_LSP object.

When a Path message signals multiple S2L sub-LSPs the path of the first S2L sub-LSP, to the egress LSR, is encoded in the ERO. The first S2L sub-LSP is the one that corresponds to the first S2L_SUB_LSP object in the Path message. The S2L sub-LSPs corresponding to the S2L_SUB_LSP objects that follow are termed as subsequent S2L sub-LSPs. One approach to encode the explicit route of a subsequent S2L sub-LSP is to include the path from the ingress to the egress of the S2L sub-LSP. However this implies potential repetition of hops that could be learned from the ERO or explicit routes of other S2L sub-LSPs. Explicit route compression using SEROs attempts to minimize such repetition and is described below.

The path of each subsequent S2L sub-LSP is encoded in a SUB_EXPLICIT_ROUTE object (SERO). The format of the SERO is the same as an ERO (as defined in [RFC3209]). Each subsequent S2L sub-LSP is represented by tuples of the form [<SUB_EXPLICIT_ROUTE>] <S2L_SUB_LSP>. There is a one to one correspondence between a S2L_SUB_LSP object and a SERO. A SERO for a particular S2L sub-LSP includes only the path from a certain branch LSR to the egress LSR if the path to that branch LSR can be derived from the ERO or other SEROs. The absence of a SERO should be interpreted as requiring hop-by-hop routing for that S2L sub-LSP. Note that the destination address is carried in the S2L sub-LSP object. The encoding of the SERO and S2L sub-LSP object are described in detail in section 20.

Explicit route compression is illustrated using the following figure.



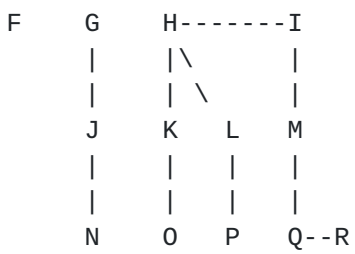


Figure 1. Explicit Route Compression

Figure 1. shows a P2MP LSP Tunnel with LSR A as the ingress LSR and six egress LSRs: (F, N, O, P, Q and R). When all the six S2L sub-LSPs are signaled in one Path message let us assume that the S2L sub-LSP to LSR F is the first S2L sub-LSP and the rest are subsequent S2L sub-LSPs. Following is one way for the ingress LSR A to encode the S2L sub-LSP explicit routes using compression:

```
S2L sub-LSP-F:   ERO = {B, E, D, C, F},  S2L_SUB_LSP Object-F
S2L sub-LSP-N:   SERO = {D, G, J, N}, S2L_SUB_LSP Object-N
S2L sub-LSP-O:   SERO = {E, H, K, O}, S2L_SUB_LSP Object-O
S2L sub-LSP-P:   SERO = {H, L, P}, S2L_SUB_LSP Object-P,
S2L sub-LSP-Q:   SERO = {H, I, M, Q}, S2L_SUB_LSP Object-Q,
S2L sub-LSP-R:   SERO = {Q, R}, S2L_SUB_LSP Object-R,
```

After LSR E processes the incoming Path message from LSR B it sends a Path message to LSR D with the S2L sub-LSP explicit routes encoded as follows:

```
S2L sub-LSP-F:   ERO = {D, C, F},  S2L_SUB_LSP Object-F
S2L sub-LSP-N:   SERO = {D, G, J, N}, S2L_SUB_LSP Object-N
```

LSR E also sends a Path message to LSR H and following is one way to encode the S2L sub-LSP explicit routes using compression:

```
S2L sub-LSP-O:   ERO = {H, K, O}, S2L_SUB_LSP Object-O
S2L sub-LSP-P:   SERO = {H, L, P}, S2L_SUB_LSP Object-P,
S2L sub-LSP-Q:   SERO = {H, I, M, Q}, S2L_SUB_LSP Object-Q,
S2L sub-LSP-R:   SERO = {Q, R}, S2L_SUB_LSP Object-R,
```

After LSR H processes the incoming Path message from E it sends a Path message to LSR K, LSR L and LSR I. The encoding for the Path message to LSR K is as follows:

```
S2L sub-LSP-O:   ERO  = {K, O}, S2L_SUB_LSP Object-O
```

The encoding of the Path message sent by LSR H to LSR L is as follows:

S2L sub-LSP-P: ERO = {L, P}, S2L_SUB_LSP Object-P,

Following is one way for LSR H to encode the S2L sub-LSP explicit routes in the Path message sent to LSR I:

S2L sub-LSP-Q: ERO = {I, M, Q}, S2L_SUB_LSP Object-Q,
S2L sub-LSP-R: SERO = {Q, R}, S2L_SUB_LSP Object-R,

The explicit route encodings in the Path messages sent by LSRs D and Q are left as an exercise to the reader.

This compression mechanism reduces the Path message size. It also reduces the processing that can result if explicit routes are encoded from ingress to egress for each S2L sub-LSP. No assumptions are placed on the ordering of the subsequent S2L sub-LSPs and hence on the ordering of the SEROs in the Path message. All LSRs need to process the ERO corresponding to the first S2L sub-LSP. A LSR needs to process a SERO for a subsequent S2L sub-LSP only if the first hop in the corresponding SERO is a local address of that LSR. The branch LSR that is the first hop of a SERO propagates the corresponding S2L sub-LSP downstream.

4. Path Message

4.1. Path Message Format

This section describes modifications made to the Path message format as specified in [RFC3209] and [RFC3473]. The Path message is enhanced to signal one or more S2L sub-LSPs. This is done by including the S2L sub-LSP descriptor list in the Path message as shown below.

```
<Path Message> ::=     <Common Header> [ <INTEGRITY> ]
                       [ [<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]
                       [ <MESSAGE_ID> ]
                       <SESSION> <RSVP_HOP>
                       <TIME_VALUES>
                       [ <EXPLICIT_ROUTE> ]
                       <LABEL_REQUEST>
                       [ <PROTECTION> ]
                       [ <LABEL_SET> ... ]
                       [ <SESSION_ATTRIBUTE> ]
                       [ <NOTIFY_REQUEST> ]
                       [ <ADMIN_STATUS> ]
                       [ <POLICY_DATA> ... ]
                       <sender descriptor>
                       [S2L sub-LSP descriptor list]
```

Following is the format of the S2L sub-LSP descriptor list.

<S2L sub-LSP descriptor list> ::= <S2L sub-LSP descriptor>
[<S2L sub-LSP descriptor list>]

<S2L sub-LSP descriptor> ::= <S2L_SUB_LSP> [<SUB_EXPLICIT_ROUTE>]

Each LSR MUST use the common objects in the Path message and the S2L sub-LSP descriptors to process each S2L sub-LSP represented by the S2L sub-LSP object and the SUB-/EXPLICIT_ROUTE object combination.

The first S2L_SUB_LSP object's explicit route is specified by the ERO. Explicit routes of subsequent S2L sub-LSPs are specified by the corresponding SERO. A SERO corresponds to the following S2L_SUB_LSP object.

The RRO in the sender descriptor contains the hops traversed by the Path message and applies to all the S2L sub-LSPs signaled in the Path message.

Path message processing is described in the next section.

4.2. Path Message Processing

The ingress-LSR initiates the set up of a S2L sub-LSP to each egress-LSR that is the destination of the P2MP LSP Tunnel. Each S2L sub-LSP is associated with the same P2MP LSP Tunnel using common P2MP SESSION object and <Source Address, LSP-ID> fields in the SENDER_TEMPLATE object. Hence it can be combined with other S2L sub-LSPs to form a P2MP LSP Tunnel. Another S2L sub-LSP belonging to the same instance of this S2L sub-LSP (i.e. the same P2MP LSP Tunnel) can share resources with this LSP. The session corresponding to the P2MP TE tunnel is determined based on the P2MP SESSION object. Each S2L sub-LSP is identified using the S2L_SUB_LSP object. Explicit routing for the S2L sub-LSPs is achieved using the ERO and SEROs.

As mentioned earlier, it is possible to signal S2L sub-LSPs for a given P2MP LSP Tunnel in one or more Path messages. And a given Path message can contain one or more S2L sub-LSPs.

4.2.1. Multiple Path messages

As described in section 3, {<EXPLICIT_ROUTE>, <S2L SUB-LSP>} or {<SUB_EXPLICIT_ROUTE>, <S2L_SUB_LSP>} tuple is used to specify a S2L sub-LSP. Multiple Path messages can be used to signal a P2MP LSP Tunnel. Each Path message can signal one or more S2L sub-LSPs. If a Path message contains only one S2L sub-LSP, each LSR along the S2L sub-LSP follows [RFC3209] procedures for processing the Path message

besides the S2L SUB-LSP object processing described in this document.

Processing of Path messages containing more than one S2L sub-LSP is described in Section 4.3.

An ingress LSR may use multiple Path messages for signaling a P2MP LSP. This may be because a single Path message may not be large enough to signal the P2MP LSP Tunnel. Or it may be while adding leaves to the P2MP LSP Tunnel the new leaves are signaled in a new Path message. Or an ingress LSR MAY choose to break the P2MP tree into separate manageable S2L sub-trees. These trees share the same root and may share the trunk and certain branches. The scope of this management decomposition of P2MP trees is bounded by a single tree and multiple S2L sub-trees with a single leaf each. As defined in [P2MP-REQ], a P2MP LSP Tunnel must have consistent attributes across all portions of a tree. This implies that each Path message that is used to signal a P2MP LSP Tunnel is signaled using the same signaling attributes with the exception of the S2L sub-LSP information.

The resulting S2L sub-LSPs from the different Path messages belonging to the same P2MP LSP Tunnel SHOULD share labels and resources where they share hops to prevent multiple copies of the data being sent.

In certain cases a transit LSR may need to generate multiple Path messages to signal state corresponding to a single received Path message. For instance ERO expansion may result in an overflow of the resultant Path message. There are two cases occurring in such circumstances, either the message can be decomposed into multiple Path messages such that each of the message carries a subset of the incoming S2L sub-LSPs carried by the incoming message, or the message can not be decomposed such that each of the outgoing Path message fits its maximum size value.

Multiple Path messages generated by a LSR that signal state for the same P2MP LSP are signaled with the same SESSION object and have the same <Source address, LSP-ID> in the SENDER_TEMPLATE object. In order to disambiguate these Path messages a <Sub-Group Originator ID, sub-Group ID> tuple is introduced (also referred to as the Sub-Group field). Multiple Path messages generated by a LSR to signal state for the same P2MP LSP have the same Sub-Group Originator ID and have a different sub-Group ID. The Sub-Group Originator ID SHOULD be set to the TE Router ID of the LSR that originates the Path message. This is either the ingress LSR or a LSR which re-originates the Path message with its own Sub-Group Originator ID. Cases when a transit LSR may change the Sub-Group Originator ID of an incoming Path message are described below. The <Sub-Group Originator ID, sub-Group ID> tuple is network-wide unique. The sub-Group ID space is specific to the Sub-Group Originator ID. Therefore the combination <Sub-Group

Originator ID, sub-Group ID> is network-wide unique. Also, a router that changes the Sub-Group Originator ID MUST use the same Sub-Group Originator ID on all Path messages for the same P2MP LSP Tunnel and SHOULD not vary the value during the life of the P2MP LSP Tunnel.

Note: This version of the document assumes that these additional fields, i.e. <Sub-Group Originator ID, sub-Group ID>, are part of the SENDER_TEMPLATE object.

4.2.2. Multiple S2L Sub-LSPs in one Path message

The S2L sub-LSP descriptor list allows the signaling of one or more S2L sub-LSPs in one Path message. It is possible to signal multiple S2L sub-LSP objects and ERO/SERO combinations in a single Path message. Note that these objects are the ones that differentiate a S2L sub-LSP. Each LSR can use the common objects in the Path message and the S2L sub-LSP descriptors to process each S2L sub-LSP.

All LSRs need to process the ERO corresponding to the first S2L sub-LSP when the ERO is present. If one or more SEROs are present an ERO MUST be present. The signaling information for the first S2L sub-LSP is propagated in a Path message by each LSR along the explicit route specified by the ERO. A LSR needs to process a S2L sub-LSP descriptor for a subsequent S2L sub-LSP only if the first hop in the corresponding SERO is a local address of that LSR. If this is not the case the S2L sub-LSP descriptor is included in the Path message sent to LSR that is the next hop to reach the first hop in the SERO. This next hop is determined by using the ERO or other SEROs that encode the path to the SERO's first hop. If this is the case and the LSR is also the egress the S2L sub-LSP descriptor is not propagated downstream. If this is the case and the LSR is not the egress the S2L sub-LSP descriptor is included in a Path message sent to the next-hop determined from the SERO. Hence a branch LSR only propagates the relevant S2L sub-LSP descriptors on each downstream link. A S2L sub-LSP descriptor that is propagated on a downstream link only contains those S2L sub-LSPs that are routed using that link. This processing may result in a subsequent S2L sub-LSP in an incoming Path message to become the first S2L sub-LSP in an outgoing Path message.

Note that if one or more SEROs contains loose hops, expansion of such loose hops may result in overflowing the Path message size. Section 4.2.3 describes how signaling of the set of S2L sub-LSPs can be split in more than one Path message.

The Record Route Object (RRO) contains the hops traversed by the Path message and applies to all the S2L sub-LSPs signaled in the Path message. A transit LSR appends its address in an incoming RRO and propagates it downstream. A branch LSR forms a new RRO for each of

the outgoing Path messages. Each such updated RRO is formed using the rules in [RFC3209].

If a LSR is unable to support a S2L sub-LSP setup, a PathErr message MUST be sent for the impacted S2L sub-LSP, and normal processing of the rest of the P2MP LSP Tunnel SHOULD continue. The default behavior is that the remainder of the LSP is not impacted (that is, all other branches are allowed to set up) and the failed branches are reported in PathErr messages in which the Path_State_Reomved flag MUST NOT be set. However, the ingress LSR may set a LSP Integrity flag (see section 21.3) to request that if there is a setup failure on any branch the entire LSP should fail to set up.

4.2.3. Transit Fragmentation

In certain cases a transit LSR may need to generate multiple Path messages to signal state corresponding to a single received Path message. For instance ERO expansion may result in an overflow of the resultant Path message. It is desirable not to rely on IP fragmentation in this case. In order to achieve this, the multiple Path messages generated by the transit LSR, MUST be signaled with the Sub-Group Originator ID set to the TE Router ID of the transit LSR and a distinct sub-Group ID. Thus each distinct Path message that is generated by the transit LSR for the P2MP LSP Tunnel carries a distinct <Sub-Group Originator ID, Sub-Group ID> tuple.

When multiple Path messages are used by an ingress or transit node, each Path message SHOULD be identical with the exception of the S2L sub-LSP related information (e.g., SERO), message and hop information (e.g., INTEGRITY, MESSAGE_ID and RSVP_HOP), and the SENDER_TEMPLATE objects. Except when performing a make-before-break operation, the tunnel sender address and LSP ID fields MUST be the same in each message, and for transit nodes, the same as the values in the Path message.

As described above one case in which the Sub-Group Originator ID of a received Path message is changed is that of transit fragmentation. The Sub-Group Originator ID of a received Path message may also be changed in the outgoing Path message and set to that of the LSR originating the Path message based on a local policy. For instance a LSR may decide to always change the Sub-Group Originator ID while performing ERO expansion. The Sub-Group ID MUST not be changed if the Sub-Group Originator ID is not being changed.

4.3. Grafting

The operation of adding egress LSR(s) to an existing P2MP LSP Tunnel is termed grafting. This operation allows egress nodes to join a P2MP LSP Tunnel at different points in time.

4.3.1. Addition of S2L Sub-LSPs

There are two methods to add S2L sub-LSPs to a P2MP LSP Tunnel. The first is to add new S2L sub-LSPs to the P2MP LSP Tunnel by adding them to an existing Path message and refreshing the entire Path message. Path message processing described in section 4 results in adding these S2L sub-LSPs to the P2MP LSP Tunnel. Note that as a result of adding one or more S2L sub-LSPs to a Path message the ERO compression encoding may have to be recomputed.

The second is to use incremental updates described in section 11.1. The egress LSRs can be added/removed by signaling only the impacted S2L sub-LSPs in a new Path message. Hence other S2L sub-LSPs do not have to be re-signaled.

5. Resv Message

5.1. Resv Message Format

The Resv message follows the [RFC3209] and [RFC3473] format:

```
<Resv Message> ::=    <Common Header> [ <INTEGRITY> ]
                      [ [<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ... ]
                      [ <MESSAGE_ID> ]
                      <SESSION> <RSVP_HOP>
                      <TIME_VALUES>
                      [ <RESV_CONFIRM> ]  [ <SCOPE> ]
                      [ <NOTIFY_REQUEST> ]
                      [ <ADMIN_STATUS> ]
                      [ <POLICY_DATA> ... ]
                      <STYLE> <flow descriptor list>


<flow descriptor list> ::= <FF flow descriptor list>
                         | <SE flow descriptor>


<FF flow descriptor list> ::= <FF flow descriptor>
                            | <FF flow descriptor list>
                            <FF flow descriptor>
```

```
<SE flow descriptor> ::= <FLOWSPEC> <SE filter spec list>

<SE filter spec list> ::= <SE filter spec>
                        | <SE filter spec list> <SE filter spec>
```

The FF flow descriptor and SE filter spec are modified as follows to identify the S2L sub-LSPs that they correspond to:

```
<FF flow descriptor> ::= [ <FLOWSPEC> ] <FILTER_SPEC> <LABEL>
                         [ <RECORD_ROUTE> ]
                         [ <S2L sub-LSP descriptor list> ]

<SE filter spec> ::=     <FILTER_SPEC> <LABEL> [ <RECORD_ROUTE> ]
                         [ <S2L sub-LSP descriptor list> ]
```

FILTER_SPEC is defined in section 20.4.

The S2L sub-LSP descriptor has the same format as in section 4.1 with the difference that a SUB_RECORD_ROUTE object is used in place of a SUB_EXPLICIT_ROUTE object.

```
<S2L sub-LSP filte descriptor list> ::= <S2L sub-LSP filter
descriptor>
                                     [ <S2L sub-LSP filter descriptor
list> ]

<S2L sub-LSP filte descriptor> ::= <S2L_SUB_LSP> [ <SUB_RECORD_ROUTE>
]
```

The SUB_RECORD_ROUTE objects follow the same compression mechanism as the SUB_EXPLICIT_ROUTE objects. Note that a Resv message can signal multiple S2L sub-LSPs that may belong to the same FILTER_SPEC object or different FILTER_SPEC objects. The same label is allocated if the FILTER_SPEC object is the same.

However different upstream labels are allocated if the <Source Address, LSP-ID> of the FILTER_SPEC object is different as that implies different P2MP LSP Tunnels.

5.2. Resv Message Processing

The egress LSR follows normal RSVP procedures while originating a Resv message. The Resv message carries the label allocated by the egress LSR.

A subsequent node allocates its own label and passes it upstream in the Resv message. The node may combine multiple flow descriptors, from different Resv messages received from downstream, in one Resv

message sent upstream. A Resv message is not sent upstream by a transit LSR until at least one Resv message has been received from a downstream neighbor except when the integrity bit is set in the LSP_ATTRIBUTE object.

Each FF flow descriptor or SE filter spec sent upstream in a Resv message includes a S2L sub-LSP descriptor list. Each such FF flow descriptor or SE filter spec for the same P2MP LSP Tunnel (whether on one or multiple Resv messages) is allocated the same label.

This label is associated by that node with all the labels received from downstream Resv messages for that P2MP LSP Tunnel. Note that a transit node may become a replication point in the future when a branch is attached to it. Hence this results in the setup of a P2MP LSP Tunnel from the ingress-LSR to the egress LSRs.

The ingress LSR may need to understand when all desired egresses have been reached. This is achieved using <S2L_SUB_LSP> objects.

Each branch node can potentially send one Resv message upstream for each of the downstream receivers. This may result in overflowing the Resv message, particularly when considering that the number of messages increases the closer the branch node is to the ingress.

Transit nodes MUST replace the Sub-Group ID fields received in the FILTER_SPEC objects with the value that was received in the Sub-Group ID field of the Path message from the upstream neighbor, when the node set the Sub-Group Originator field in the associated Path message. ResvErr message generation is unmodified. Nodes propagating a received ResvErr message MUST use the Sub-Group field values carried in the corresponding Resv message.

The solution for this issue is for further discussion.

5.2.1. Resv Message Throttling

A branch node needs to send the Resv message being sent upstream whenever there is a change in a Resv message for a S2L sub-LSP received from downstream. This can result in excessive Resv messages sent upstream, particularly when the S2L sub-LSPs are established for the first time. In order to mitigate this situation, branch nodes MAY limit their transmission of Resv messages. Specifically, in the case where the only change being sent in a Resv message is in one or more SRRO objects, the branch node SHOULD transmit the Resv message only after a delay time has passed since the transmission of the previous Resv message for the same session. This delayed Resv message SHOULD include SRROs for all branches. Specific mechanisms for Resv message throttling are implementation dependent and are outside the

scope of this document.

5.3. Record Routing

5.3.1. RRO Processing

A Resv message contains a recorded route per S2L sub-LSP that is being signaled by the Resv message if the sender node requests route recording by including a RRO in the Path message. The same rule is used during signaling of P2MP LSP Tunnels. Thus insertion of the RRO in the Path message used to signal one or more S2L sub-LSPs triggers the inclusion of an RRO for each sub-LSP signaled in that Path message or any derivative Path message.

The record route of the first S2L sub-LSP is encoded in the RRO. Additional RROs for the subsequent S2L sub-LSPs are referred to as SUB_RECORD_ROUTE objects (SRROs). Their format is specified in section 20.6. The ingress node then receives the RRO and possibly the SRRO corresponding to each subsequent S2L sub-LSP. Each S2L_SUB_LSP object is followed by the RRO/SRRO. The ingress node can then determine the recorded route corresponding to a particular S2L sub-LSP. The RRO and SRROs can be used to construct the end-to-end Path for each S2L sub-LSP.

6. Reservation Style

TBD

7. PathTear Message

7.1. PathTear message Format

The format of the PathTear message is as follows:

```
<PathTear Message> ::= <Common Header> [ <INTEGRITY> ]
                       [ [ <MESSAGE_ID_ACK> |
                         <MESSAGE_ID_NACK> ... ]
                       [ <MESSAGE_ID> ]
                       <SESSION> <RSVP_HOP>
                       [ <sender descriptor> ]
                       [ <S2L sub-LSP descriptor list> ]

              <sender descriptor> ::= (see earlier definition)
```

Note: it is assumed that the S2L sub-LSP descriptor will not include the SUB_EXPLICIT_ROUTE object associated with each S2L_SUB_LSP being

deleted

7.2. Pruning

The operation of removing egress LSR(s) from an existing P2MP LSP Tunnel is termed pruning. This operation allows egress nodes to leave a P2MP LSP Tunnel at different points in time. This section describes various mechanisms to perform pruning. Further discussion and feedback is needed to finesse these mechanisms.

7.2.1. Explicit S2L Sub-LSP Teardown

The S2L sub-LSP(s) being removed from the P2MP LSP Tunnel are signaled in a PathTear message. The PathTear message includes the S2L sub-LSP descriptor list which is included before the sender descriptor. Note that the PathTear message contains only the S2L sub-LSP(s) being removed and rest of the P2MP LSP Tunnel does not have to be re-signaled. This results in removal of the state corresponding to these S2L sub-LSPs. State for rest of the S2L sub-LSPs is not modified.

In the first mechanism in order to delete one or more S2L Sub-LSPs, a PathTear message is sent with the list of S2L sub-LSPs being deleted. This is a form of explicit tear down. A single PathTear message can only contain S2L sub-LSPs that were signaled by the ingress using the same <Sub-Group Originator ID, Sub-Group ID> tuple. The PathTear message is signaled with the SESSION and SENDER_TEMPLATE objects corresponding to the P2MP LSP Tunnel and the <Sub-Group Originator ID, Sub-Group ID> tuple corresponding to the S2L sub-LSPs that are being deleted. A transit LSR that propagates the PathTear message downstream MUST ensure that it sets the <Sub-Group Originator ID, Sub-Group ID> tuple in the PathTear message to the values used to generate the last Path message that corresponds to the S2L sub-LSPs signaled in the PathTear message that it generates. The transit LSR may need to generate multiple PathTear messages for an incoming PathTear message if it had performed transit fragmentation for the corresponding incoming Path message.

The Path messages from which the S2L sub-LSPs were deleted need to be refreshed with the remaining S2L sub-LSPs. Note that as a result of deleting one or more S2L sub-LSPs from a Path message the ERO compression encoding may have to be recomputed.

When the last S2L sub-LSP is to be removed from a Path state, i.e., there are no remaining S2L sub-LSPs to send in a Path message, a PathTear message SHOULD be sent carrying the Sub-Group ID of the Path message that no longer has any S2L sub-LSPs.

The second mechanism is an explicit teardown mechanism that defines new syntax and semantics for a PathTear message. This new mechanism minimizes signaling required to remove a subset of S2L sub-LSPs set signaled in a Path message, and thereby reduces associated processing. When using this mechanism each identified S2L sub-LSP is removed from the P2MP LSP Tunnel state, even if the S2L sub-LSP is advertised in multiple Path message.

When using this approach, a PathTear message is generated. The PathTear message MUST identify each S2L sub-LSP to be removed, via a S2L_SUB_LSP object per S2L Sub-LSP, and include a SENDER_TEMPLATE object corresponding to the Path state being modified. The Sub-Group ID valued contained in the SENDER_TEMPLATE object message MUST be set to zero (0). Subsequent Path messages associated with the P2MP LSP Tunnel MUST NOT contain the removed S2L sub-LSPs, unless that S2L sub-LSP is being re-added to the P2MP LSP.

To support the second mechanism, the receiver of PathTear message that is associated with a P2MP LSP Tunnel MUST check the value of a received Sub-Group ID fields. When there is no SENDER_TEMPLATE object present or the value of the Sub-Group ID fields is non-zero, then PathTear processing as defined in the above explicit tear down mechanism must be followed. When the Sub-Group ID field is zero (0), then the processing node MUST remove the identified egresses from all control plane state associated with the P2MP LSP Tunnel and adjust the data path appropriately.

7.2.2. Implicit S2L Sub-LSP Teardown

The third mechanism to delete S2L sub-LSPs is implicit teardown which uses standard RSVP message processing. Per standard RSVP processing, a S2L sub-LSP may be removed from a P2MP TE LSP by sending a modified message for the Path or Resv message that previously advertised the S2L sub-LSP. This message MUST list all S2L sub-LSPs that are not being removed. When using this approach, a node processing a message that removes a S2L sub-LSP from a P2MP TE LSP MUST ensure that the S2L sub-LSP is not included in any other Path state associated with session before interrupting the data path to that egress. All other message processing remains unchanged.

7.2.3. P2MP TE LSP Teardown

This operation is accomplished by listing all the S2L sub-LSPs in a PathTear message.

A PathTear message must be generated for each Path message used to signal the P2MP LSP Tunnel.

8. Notify and ResvConf Messages

Notify messages, see [RFC3473], may contain either SENDER_TEMPLATE or FILTER_SPEC objects, but are sent in a targeted fashion. This means that the Sub-Group fields cannot be updated in transit and is unlikely to provide any value to the Notify message recipient. Therefore, the receiver of a Notify message MUST identify the sender state referenced in the message based on the Source address and LSP ID contained in the received SENDER_TEMPLATE or FILTER_SPEC objects rather than, as is normally done, based on the whole objects.

ResvConf messages may contain FILTER_SPEC objects and may also be sent in a targeted fashion. As with Notify messages, the receiver of a ResvConf message MUST identify the state referenced in the message based on the address and LSP ID contained in the received FILTER_SPEC object rather than, as is normally done, based on the whole objects.

9. Error Processing

Note that a LSR on receiving a PathErr/ResvErr message for a particular S2L sub-LSP changes the state only for that S2L sub-LSP. Hence other S2L sub-LSPs are not impacted. In case the ingress node requests the maintenance of the 'LSP Integrity', any error reported within the P2MP TE LSP must be reported at (least at) any other branching nodes belonging to this LSP. Therefore, reception of an error message for a particular S2L sub-LSP MAY change the state of any other S2L sub-LSP of the same P2MP TE LSP.

9.1. PathErr Message Format

A PathErr message will include one or more S2L_SUB_LSP objects. The resulting modified format of a PathErr Message is:

```
<PathErr Message> ::=    <Common Header> [ <INTEGRITY> ]
                          [ [<MESSAGE_ID_ACK> |
                            <MESSAGE_ID_NACK>] ... ]
                          [ <MESSAGE_ID> ]
                          <SESSION> <ERROR_SPEC>
                          [ <ACCEPTABLE_LABEL_SET> ... ]
                          [ <POLICY_DATA> ... ]
                          <sender descriptor>
                          [ <S2L sub-LSP descriptor list> ]
```

PathErr messages generation is unmodified, but nodes that set the Sub-Group Originator field and propagate a received PathErr message upstream MUST replace the Sub-Group fields received in the PathErr message with the value that was received in the Sub-Group fields of

the Path message from the upstream neighbor. Note the receiver of a PathErr message is able to identify the errored outgoing Path message, and outgoing interface, based on the Sub-Group fields received in the error message.

9.2. Handling of Failures at Branch LSRs

During setup and during normal operation, PathErr messages may be received at a branch node. In all cases, a received PathErr message is first processed per standard processing rules. That is, the PathErr message is sent hop-by-hop to the ingress/branch LSR for that Path message. Intermediate nodes until this ingress/branch LSR MAY inspect this message but take no action upon it. The behavior of a branch LSR that generates a PathErr message is under the control of the ingress LSR.

The default behavior is that the PathErr does not have the Path_State_Removed flag set. However, if the ingress LSR has set the 'LSP Integrity' flag on the Path message (see LSP_ATTRIBUTE object in section 21.3) and if the Path_State_Removed flag is supported, the LSR generating a PathErr to report the failure of a branch of the P2MP LSP Tunnel SHOULD set the Path_State_Removed flag.

A branch LSR that receives a PathErr message with the Path_State_Removed flag clear MUST act according to the wishes of the ingress LSR. The default behavior is that the branch LSR forwards the PathErr upstream and takes no further action. However, if the LSP integrity flag is set on the Path message, the branch LSR MUST send PathTear on all downstream branches and send the PathErr upstream with the Path_State_Removed flag set (per [RFC3473]).

In all cases, the PathErr message forwarded by a branch LSR MUST contain the S2L sub-LSP identification and explicit routes of all branches that are errored (reported by received PathErr messages) and all branches that are explicitly torn by the branch LSR.

10. Refresh Reduction

The refresh reduction procedures described in [RFC2961] are equally applicable to P2MP LSP Tunnels described in this document. Refresh reduction applies to individual messages and the state they install/maintain, and that continues to be the case for P2MP LSP Tunnels.

11. State Management

State signaled by a P2MP Path message is managed by an implementation using the <P2MP ID, Tunnel ID, Extended Tunnel ID> as part of the SESSION object and <Tunnel Sender Address, LSP ID, Sub-Group Originator ID, Sub-Group ID> as part of the SENDER_TEMPLATE object.

Additional information signaled in the Path message is part of the state created by an implementation. This mandatorily includes PHOP and SENDER_TSPEC objects.

11.1. Incremental State Update

RSVP as defined in [RFC2205] and as extended by RSVP-TE [RFC3209] and GMPLS [RFC3473] uses the same basic approach to state communication and synchronization, namely full state is sent in each state advertisement message. Per [RFC2205] Path and Resv messages are idempotent. Also, [RFC2961] categorizes RSVP messages into two types: trigger and refresh messages and improves RSVP message handling and scaling of state refreshes but does not modify the full state advertisement nature of Path and Resv messages. The full state advertisement nature of Path and Resv messages has many benefits, but also has some drawbacks. One notable drawback is when an incremental modification is being made to a previously advertised state. In this case, there is the message overhead of sending the full state and the cost of processing it. It is desirable to overcome this drawback and add/delete S2L sub-LSPs to a P2MP LSP Tunnel by incrementally updating the existing state.

It is possible to use the procedures described in this document to allow S2L sub-LSPs to be incrementally added or deleted from the P2MP LSP by allowing a Path or a PathTear message to incrementally change the existing P2MP LSP Tunnel Path state.

As described in section 4.2, multiple Path messages can be used to signal a P2MP LSP Tunnel. The Path messages are distinguished by different <Sub-Group Originator ID, Sub-Group ID> tuples in the SENDER_TEMPLATE object. In order to perform incremental S2L sub-LSP state addition a separate Path message with a new sub-Group ID is

used to add the new S2L sub-LSPs, by the ingress LSR. The Sub-Group Originator ID MUST be set to the TE Router ID [RFC3477] of the node that sets the Sub-Group ID.

This maintains the idempotent nature of RSVP Path messages; avoids keeping track of individual S2L sub-LSP state expiration and provides the ability to perform incremental P2MP LSP Tunnel state updates.

11.2. Combining Multiple Path Messages

There is a tradeoff between the number of Path messages used by the ingress to maintain the P2MP LSP Tunnel and using full state refresh to add S2L sub-LSPs. It is possible to combine S2L sub-LSPs previously advertised in different Path messages into a single Path message in order to reduce the number of Path messages needed to maintain the P2MP LSP. This can also be done by a transit node that performed fragmentation and at a later point is able to combine multiple Path messages that it generated into a single Path message. This may happen when one or more S2L sub-LSPs are pruned from the existing Path states.

The new Path message is signaled by the node that is combining multiple Path messages with all the S2L sub-LSPs that are being combined in a single Path message. This Path message contains a new Sub-Group ID field value. When a new Path and Resv message that is signaled for an existing S2L sub-LSP is received by a transit LSR, state including the new instance of the S2L sub-LSP is created.

The S2L sub-LSP SHOULD continue to be advertised in both the old and new Path messages until a Resv message listing the S2L sub-LSP and corresponding to the new Path message is received by the combining node. Hence until this point state for the S2L sub-LSP SHOULD be maintained as part of the Path state for both the old and the new Path message [Section 3.1.3, 2205]. At that point the S2L sub-LSP SHOULD be deleted from the old Path state using a PathTear message. The S2L sub-LSP should also be removed from the old Path message and the old Path message should be signaled again, if there are other remaining S2L sub-LSPs in the old Path message.

A Path message with a Sub-Group_ID(n+1) may signal a set of S2L sub-LSPs that belong partially or entirely to an already existing Sub-Group_ID(i), i <= n, the SESSION object and <Sender Tunnel Address, LSP-ID, Sub-Group Originator ID> being the same. Or it may signal a strictly non-overlapping new set of S2L sub-LSPs with a strictly higher Sub-Group_ID value.

1) If Sub-Group_ID(i) = Sub-Group_ID(n+1), i =< n then either a full refresh is indicated by the Path message or a S2L Sub-LSP is added

to/deleted from the group signaled by Sub-Group_ID(n+1)

2) If Sub-Group_ID(i) != Sub-Group_ID(n+1), i =< n then the Path message is signaling a set of S2L sub-LSPs that belong partially or entirely to an already existing Sub-Group_ID(i) or a strictly non-overlapping set of S2L sub-LSPs.

12. Control of Branch Fate Sharing

An ingress LSR can control the behavior of an LSP if there is a failure during LSP setup or after an LSP has been established. The default behavior is that only the branches downstream of the failure are not established, but the ingress may request 'LSP integrity' such that any failure anywhere within the LSP tree causes the entire P2MP LSP Tunnel to fail.

The ingress LSP may request 'LSP integrity' by setting bit [section 21.3] of the Attributes Flags TLV. The bit is set if LSP integrity is required.

It is RECOMMENDED to use the LSP_ATTRIBUTES Object for this flag and not the LSP_REQUIRED_ATTRIBUTES Object.

A branch LSR that supports the Attributes Flags TLV and recognizes this bit MUST support LSP integrity or reject the LSP setup with a PathErr carrying the error "Routing Error"/"Unsupported LSP Integrity"

13. Admin Status Change

A branch node that receives an ADMIN_STATUS object processes it normally and also relays the ADMIN_STATUS object in a Path on every branch. All Path messages may be concurrently sent to the downstream neighbors.

Downstream nodes process the change in the ADMIN_STATUS object per [RFC3473], including generation of Resv messages. When the last received upstream ADMIN_STATUS object had the R bit set, branch nodes wait for a Resv message with a matching ADMIN_STATUS object to be received (or a corresponding PathErr or ResvTear messsage) on all branches before relaying a corresponding Resv message upstream.

14. Label Allocation on LANs with Multiple Downstream Nodes

A sender on a LAN uses a different label for sending traffic to each node on the LAN that belongs to the P2MP LSP Tunnel. Thus the sender performs replication. It may be considered desirable on a LAN to use the same label for sending traffic to multiple nodes belonging to the same P2MP LSP Tunnel, to avoid replication. Procedures for doing this are for further study. Given the relatively small number of receivers on LANs typically deployed in MPLS networks, this is not currently seen as a practical problem. Furthermore avoiding replication at the sender on a LAN requires significant complexity in the control plane. Given the tradeoff we propose the use of replication by the sender on a LAN.

15. Make-before-break

Let's consider the following cases where make-before-break is needed:

15.1. P2MP Tree Re-optimization

In this case all the S2L sub-LSPs are signaled with a different LSP ID by the ingress-LSR and follow make-before-break procedure [RFC3209]. Thus a new P2MP LSP Tunnel instance is established. Each S2L sub-LSP is signaled with a different LSP ID, corresponding to the new P2MP TE LSP. The ingress can, after moving traffic to the new instance, tear down the previous P2MP LSP Tunnel instance.

15.2. Re-optimization of a subset of S2L sub-LSPs

One way to accomplish re-optimization of a subset of S2L sub-LSPs that belong to a P2MP LSP Tunnel is to resignal the entire tree with a new LSP-ID as described in the previous subsection.

(There is NO-CONSENSUS between the authors on rest of the text in this subsection and it needs further discussion.)

It is possible to accomplish re-optimization of one or more S2L sub-LSPs without re-signaling rest of the P2MP LSP. To achieve this a sub-LSP ID is used to identify each S2L sub-LSP. This is encoded in the S2L sub-LSP object. Each re-optimized S2L sub-LSP is signaled with a different sub-LSP ID and hence a new S2L sub-LSP is established. Once the new setup is complete, the old S2L sub-LSP can be torn down. In some cases this may result in transient data duplication.

16. Fast Reroute

[RSVP-FR] extensions can be used to perform fast reroute for the mechanism described in this document.

16.1. Facility Backup

Facility backup as described in [RSVP-FR] can be used to protect P2MP LSP Tunnels.

If link protection is desired, a bypass tunnel is used to protect the link between the PLR and next-hop. Thus all S2L sub-LSPs that use the link can be protected in the event of link failure. Note that all such S2L sub-LSPs belonging to a particular instance of a P2MP tunnel will share the same outgoing label on the link between the PLR and the next-hop. This is the P2MP LSP label on the link. Label stacking is used to send data for each P2MP LSP in the bypass tunnel. The inner label is the P2MP LSP Tunnel label allocated by the nhop. During failure Path messages for each S2L sub-LSP, that is effected, will be sent to the MP, by the PLR. It is recommended that the PLR use the sender template specific method to identify these Path messages. Hence the PLR will set the source address in the sender template to a local PLR address. The MP will use the LSP-ID to identify the corresponding S2L sub-LSPs.

The MP MUST not use the <sub-group originator ID, sub-group ID> while identifying the corresponding S2L sub-LSPs.

In order to further process a S2L sub-LSP it will determine the protected S2L sub-LSP using the LSP-id and the S2L sub-LSP object.

If node protection is desired, the bypass tunnel must intersect the path of the protected S2L sub-LSPs somewhere downstream of the PLR. This constrains the set of S2L sub-LSPs being backed-up via that bypass tunnel to those that pass through a common downstream MP. The MP will allocate the same label to all such S2L sub-LSPs belonging to a particular instance of a P2MP tunnel. This will be the inner label used during label stacking. This may require the PLR to be branch capable as multiple bypass tunnels may be required to backup the set of S2L sub-LSPs passing through the protected node. Else all the S2L sub-LSPs being backed up must pass through the same MP.

16.2. One to One Backup

One to one backup as described in [RSVP-FR] can be used to protect a particular S2L sub-LSP against link and next-hop failure. Protection may be used for one or more S2L sub-LSPs between the PLR and the next-hop. All the S2L sub-LSPs corresponding to the same instance of

the P2MP tunnel, between the PLR and the next-hop share the same P2MP LSP Tunnel label.

All or some of these S2L sub-LSPs may be protected.

The detour S2L sub-LSPs may or may not share labels, depending on the detour path. Thus the set of outgoing labels and next-hops for a P2MP LSP Tunnel that was using a single next-hop and label between the PLR and next-hop before protection, may change once protection is triggerred.

Its is recommended that the path specific method be used to identify a backup S2L sub-LSP. Hence the DETOUR object will be inserted in the backup Path message. A backup S2L sub-LSP MUST be treated as belonging to a different P2MP tunnel instance than the one specified by the LSP-id. Furthermore multiple backup S2L sub-LSPs MUST be treated as part of the same P2MP tunnel instance if they have the same LSP-id and the same DETOUR objects. Note that as specified in section 3 S2L sub-LSPs between different P2MP tunnel instances use different labels.

If there is only S2L sub-LSP in the Path message, the DETOUR object applies to that sub-LSP. If there are multiple S2L sub-LSPs in the Path message the DETOUR applies to all the S2L sub-LSPs.

17. Support for LSRs that are not P2MP Capable

It may be that some LSRs in a network are capable of processing the P2MP extensions described in this document, but do not support P2MP branching in the data plane. If such an LSR is requested to become a branch LSR by a received Path message, it MUST respond with a PathErr message carrying the Error Value "Routing Error" and Error Code "Unable to Branch".

Its also conceivable that some LSRs, in a network deploying P2MP capability, may not support the extensions described in this document. If a Path message for the establishment of a P2MP LSP Tunnel reaches such an LSR it will reject it with a PathErr because it will not recognize the C-Type of the P2MP SESSION object.

LSRs that do not support the P2MP extensions in this document may be included as transit LSRs by the use of LSP-stitching and LSP-hierarchy [LSP-HIER]. Note that LSRs that are required to play any other role in the network (ingress, branch or egress) MUST support the extensions defined in this document.

The use of LSP-stitching and LSP-hierarchy [LSP-HIER] allows P2MP LSP

Tunnels to be built in such an environment. A P2P LSP segment is signaled from the previous P2MP capable hop of a legacy LSR to the next P2MP capable hop. Of course this assumes that intermediate legacy LSRs are transit LSRs and cannot act as P2MP branch points. Transit LSRs along this LSP segment do not process control plane messages associated with a P2MP LSP Tunnel. Furthermore these LSRs also do not need to have P2MP data plane capability as they only need to process data belonging to the P2P LSP segment. Hence these LSRs do not need to support P2MP MPLS. This P2P LSP segment is stitched to the incoming P2MP LSP Tunnel. After the P2P LSP segment is established the P2MP Path message is sent to the next P2MP capable LSR as a directed Path message. The next P2MP capable LSR stitches the P2P LSP segment to the outgoing P2MP LSP Tunnel.

In packet networks, the S2L sub-LSPs may be nested inside the outer P2P LSP Tunnel. Hence label stacking can be used to enable use of the same LSP Tunnel segment for multiple P2MP LSP Tunnels. Stitching and nesting considerations and procedures are described further in [INT-REG].

It may be an overhead for an operator to configure the P2P LSP segments in advance, when it is desired to support legacy LSRs. It may be desirable to do this dynamically. The ingress can use IGP extensions to determine non P2MP capable LSRs. It can use this information to compute S2L sub-LSP paths such that they avoid these legacy LSRs. The explicit route object of a S2L sub-LSP path may contain loose hops if there are legacy LSRs along the path. The corresponding explicit route contains a list of objects upto the P2MP capable LSR that is adjacent to a legacy LSR followed by a loose object with the address of the next P2MP capable LSR. The P2MP capable LSR expands the loose hop using its TED. When doing this it determines that the loose hop expansion requires a P2P LSP to tunnel through the legacy LSR. If such a P2P LSP exists, it uses that P2P LSP. Else it establishes the P2P LSP. The P2MP Path message is sent to the next P2MP capable LSR using non-adjacent signaling. The P2MP capable LSR that initiates the non-adjacent signaling message to the next P2MP capable LSR may have to employ a fast detection mechanism such as [BFD] to the next P2MP capable LSR.

This may be needed for the directed Path message Head-End to use node protection FRR when the protected node is the directed Path message tail.

Note that legacy LSRs along a P2P LSP segment cannot perform node protection of the tail of the P2P LSP segment.

18. Reduction in Control Plane Processing with LSP Hierarchy

It is possible to take advantage of LSP hierarchy [LSP-HIER] while setting up P2MP LSP Tunnels, as described in the previous section, to reduce control plane processing along transit LSRs that are P2MP capable. This is applicable only in environments where LSP hierarchy can be used. Transit LSRs along a P2P LSP segment, being used by a P2MP LSP Tunnel, do not process control plane messages associated with the P2MP LSP Tunnel. Infact they are not aware of these messages as they are tunneled over the P2P LSP segment. This reduces the amount of control plane processing required on these transit LSRs.

Note that the P2P LSP segments can be dynamically set up as described in the previous section or preconfigured. For example in Figure 2, PE1 can setup a P2P LSP to P1 and use that as a LSP segment. The Path messages for PE3 and PE4 can now be tunneled over the LSP segment. Thus P3 is not aware of the P2MP LSP Tunnel and does not process the P2MP control messages.

19. P2MP LSP Tunnel Remerging and Cross-Over

The functional description described so far assumes that multiple Path messages received by a LSR for the same P2MP LSP Tunnel arrive on the same incoming interface. However this may not always be the case. Further discussion is needed for this section.

P2MP tree remerging or cross-over occurs when a transit or egress node receives the signaling state i.e. Path message for the same P2MP TE LSP from more than one previous hop. If the re-merged S2L sub-LSPs are sent out on different interfaces there is no data plane issue. However if the re-merged S2L sub-LSPs are sent out on the same interface it can result in data duplication downstream. In order to describe identification of cross over and remerging by a LSR let us list the various cases when state for a S2L sub-LSP is received by a LSR.

Case1: S2L sub-LSP already exist as part of an existing Path state. The following are the various sub-cases.

a) The new S2L sub-LSP uses the same PHOP and outgoing interface as the existing S2L sub-LSP. This is either a refresh or can occur when multiple existing Path messages are combined in a new Path message.

b) The new S2L sub-LSP uses the same PHOP but different outgoing interface as the existing S2L sub-LSP. This is a case of re-routing.

c) The new S2L sub-LSP uses a different PHOP and same outgoing

interface as the existing S2L sub-LSP. This is a case of re-merging.

d) The new S2L sub-LSP uses a different PHOP and a different outgoing interface as compared to the existing S2L sub-LSP. This is a case of cross-over.

Case2: S2L sub-LSP does not exist as part of an existing Path state. The following are the sub-cases.

a) The new S2L sub-LSP uses a PHOP and outgoing interface that is same as the PHOP and outgoing interface used by an existing S2L sub-LSP. This is a legal case of signaling a new S2L sub-LSP.

b) The new S2L sub-LSP uses a PHOP that is same as that used by an existing S2L sub-LSP. However the outgoing interface is different from the outgoing interfaces used by existing S2L sub-LSPs. This is a legal case of signaling a new S2L sub-LSP.

c) The new S2L sub-LSP uses a different PHOP than that used by any of the existing S2L sub-LSP. However the outgoing interface is same as the outgoing interface used by an existing S2L sub-LSPs. This is a case of remerging.

d) The new S2L sub-LSP uses a different PHOP than that used by any of the existing S2L sub-LSP. Also the outgoing interface is different from the outgoing interfaces used by existing S2L sub-LSPs. This is a case of cross-over.

Cases 1(d) and 2(d) above identify cross-over and this is considered legal. Cases 1(c) and 2(c) above identify remerging in the data plane. If the LSR is capable of remerging in the data plane this is considered legal.

The below procedure applies for remerging.

The remerge error case is detected by checking incoming Path messages that represent new P2MP TE LSP state and seeing if they represent both known LSP state and a different S2L sub-LSP list. Specifically, the remerge check MUST be performed when processing Path messages that contain SESSION, SENDER_TEMPLATE and RSVP_HOP objects that have not previously been seen on a particular interface. The remerge check consists of attempting to locate state that has the same values in the SESSION object and in the tunnel sender address and LSP ID fields of the SENDER_TEMPLATE object.

If no matching state is located, then there is no remerge condition.

If matching state is found, then the list of S2L Sub-LSPs associated

with the new Path message is compared against the list present in the located state. If any addresses in the lists of S2L sub-LSPs match, then it is the legal LSP rerouting case mentioned here above.

If there are no overlap in the lists, and the LSR is capable of remerging in the data plane, this is considered legal. Else the new Path message MUST be handled according to remerge error processing as described below.

The LSR generates a PathErr message with Error Code "Routing Problem/P2MP Remerge Detected" towards the upstream node (i.e. the node that sent the Path message) until it reaches the node that caused the remerge condition. Identification of the offending node requires special processing by the nodes upstream of the error. A node that receives a PathErr message that contains a the error "Routing Problem/P2MP Remerge Detected" MUST check to see if it is the offending node. This check is done by comparing the S2L sub-LSPs listed in the PathErr message with existing LSP state. If any of the egresses are already present in any Path state associated with the P2MP TE LSP other than the one associated with the <SESSION, SENDER_TEMPLATE> objects signaled in the PathErr message, then the node is the signaling branch node that caused the remerge condition. This node SHOULD then correct the remerge condition by adding all S2L sub-LSPs listed in the offending Path state to the Path state (and Path message) associated to these S2L sub-LSPs. Note that the new Path state may be sent out the same outgoing interface in different Path messages in order to meet IP packet size limitations. If use of a new outgoing interface violates one or more SERO constraint, then a PathErr message containing the associated egresses and any identified valid egresses SHOULD be generated with the error code "Routing Problem" and error value of "ERO Resulted in Remerge".

This process may continue hop-by-hop until the ingress is reached. The only case where this process will fail is when all the listed S2L sub-LSPs are deleted prior to the PathErr message propagating to the ingress. In this case, the whole process will be corrected on the next (refresh or trigger) transmission of the offending Path message.

In all cases where a remerge error is not detected, normal processing continues.

20. New and Updated Message Objects

This section presents the new and updated RSVP message objects used by this document.

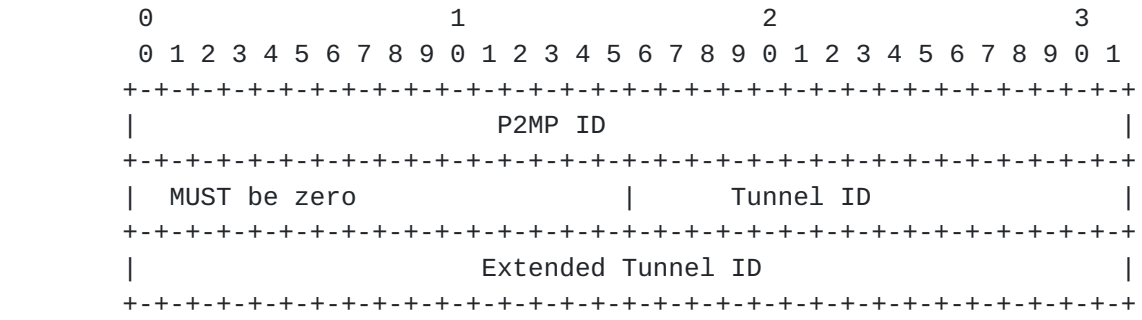
20.1. P2MP LSP Tunnel SESSION Object

A P2MP LSP Tunnel SESSION object is used. This object uses the existing SESSION C-Num. New C-Types are defined to accommodate a logical P2MP destination identifier of the P2MP Tunnel. This SESSION object has a similar structure as the existing point to point RSVP-TE SESSION object. However the destination address is set to the P2MP ID instead of the unicast Tunnel Endpoint address. All S2L sub-LSPs part of the same P2MP LSP Tunnel share the same SESSION object. This SESSION object identifies the P2MP Tunnel.

The combination of the SESSION object, the SENDER_TEMPLATE object and the S2L SUB-LSP object, identifies each S2L sub-LSP. This follows the existing P2P RSVP-TE notion of using the SESSION object for identifying a P2P Tunnel which in turn can contain multiple LSP Tunnels, each distinguished by a unique SENDER_TEMPLATE object.

20.1.1. P2MP IPv4 LSP SESSION Object

Class = SESSION, P2MP_LSP_TUNNEL_IPv4 C-Type = TBD



P2MP ID

A 32-bit identifier used in the SESSION object that remains constant over the life of the P2MP tunnel. It encodes the P2MP ID and identifies the set of destinations of the P2MP LSP Tunnel.

Tunnel ID

A 16-bit identifier used in the SESSION object that remains constant over the life of the P2MP tunnel.

Extended Tunnel ID

A 32-bit identifier used in the SESSION object that remains constant over the life of the P2MP tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-PID pair may place their IPv4 address here as a globally unique identifier [RFC3209].

20.1.2. P2MP IPv6 LSP SESSION Object

This is same as the P2MP IPv4 LSP SESSION Object with the difference that the extended tunnel ID may be set to a 16 byte identifier [RFC3209].

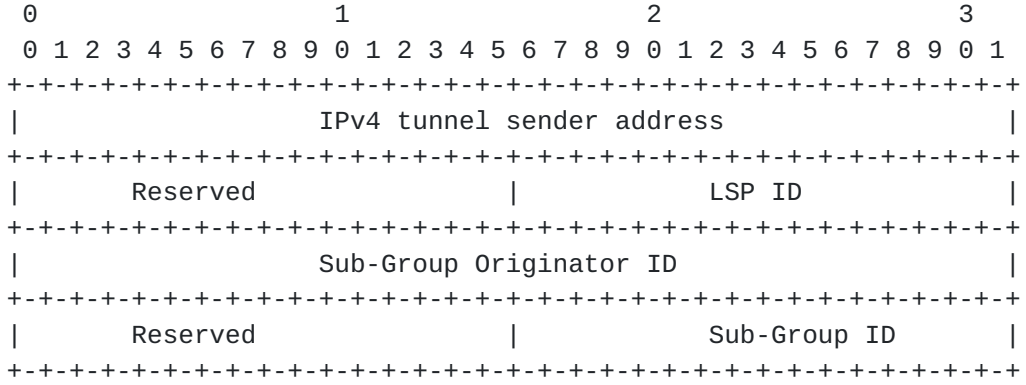
20.2. SENDER_TEMPLATE object

The sender template contains the ingress-LSR source address. LSP ID can be changed to allow a sender to share resources with itself. Thus multiple instances of the P2MP tunnel can be created, each with a different LSP ID. The instances can share resources with each other, but use different labels. The S2L sub-LSPs corresponding to a particular instance use the same LSP ID.

As described in section 4.2 it is necessary to distinguish different Path messages that are used to signal state for the same P2MP LSP Tunnel by using a <Sub-Group ID Originator ID, Sub-Group ID> tuple. There are various methods to encode this information. This document proposes the use of the SENDER_TEMPLATE object and modifies it to carry this information as shown below. This encoding is subject to review by the MPLS WG.

20.2.1. P2MP IPv4 LSP Tunnel SENDER_TEMPLATE Object

Class = SENDER_TEMPLATE, P2MP_LSP_TUNNEL_IPv4 C-Type = TBD



IPv4 tunnel sender address
See [RFC3209]

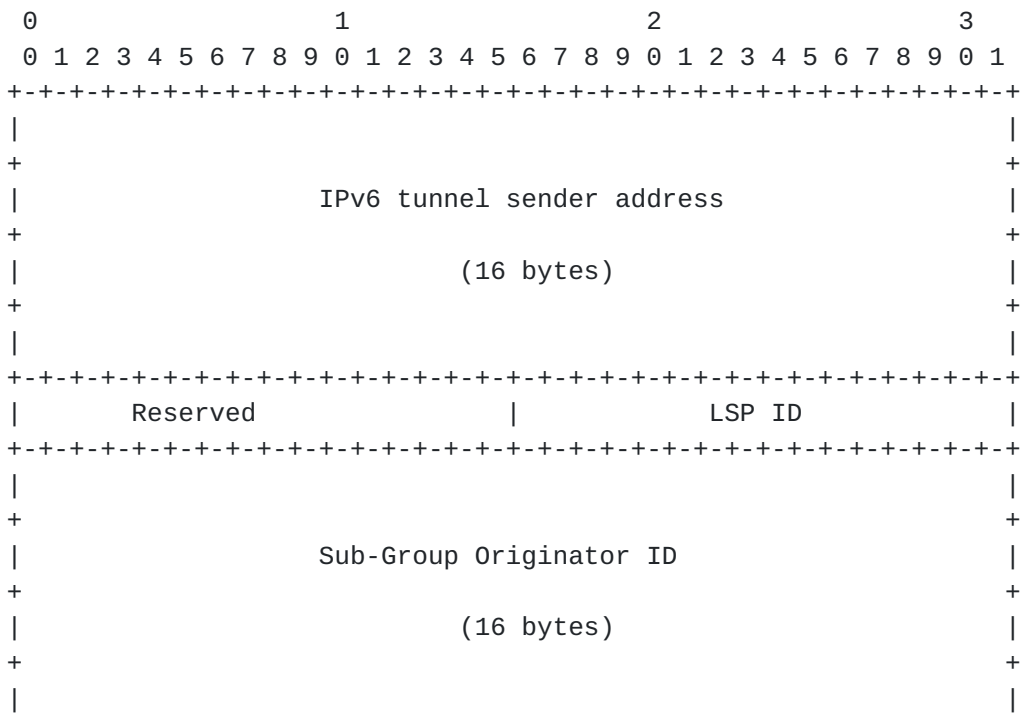
Sub-Group Originator ID
The Sub-Group Originator ID is set to the TE Router ID of the LSR that originates the Path message. This is either the ingress LSR or a LSR which re-originates the Path message with its own Sub-Group Originator ID.

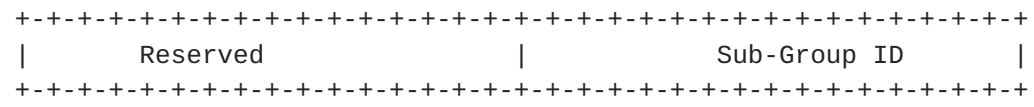
Sub-Group ID
An identifier of a Path message used to differentiate multiple Path messages that signal state for the same P2MP LSP. This may be seen as identifying a group of one or more egress nodes targeted by this Path message. If the third mechanism for pruning is used as described in section 7.2, the Sub-Group ID value of zero (0) has special meaning and MUST NOT be used with P2MP LSP Tunnels in messages other than PathTear messages. Use of a Sub-Group ID value of zero (0) in PathTear messages is defined below.

LSP ID
See [RFC3209]

20.2.2. P2MP LSP Tunnel IPv6 SENDER_TEMPLATE Object

Class = SENDER_TEMPLATE, P2MP_LSP_TUNNEL_IPv6 C-Type = TBD





IPv6 tunnel sender address
See [RFC3209]

Sub-Group Originator ID
The Sub-Group Originator ID is set to the IPv6 TE Router ID of the LSR that originates the Path message. This is either the ingress LSR or a LSR which re-originates the Path message with its own Sub-Group Originator ID.

Sub-Group ID
As above.

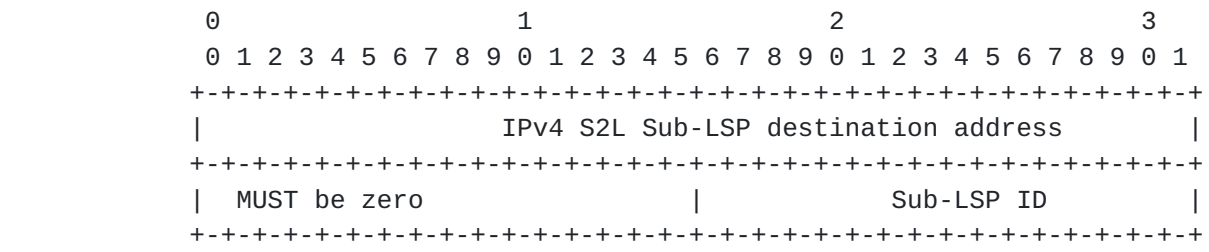
LSP ID
See [RFC3209]

20.3. S2L SUB-LSP IPv4 Object

A new S2L Sub-LSP object identifies a particular S2L sub-LSP belonging to the P2MP LSP Tunnel.

20.3.1. S2L SUB-LSP IPv4 Object

SUB_LSP Class = TBD, S2L_SUB_LSP_IPv4 C-Type = TBD



IPv4 Sub-LSP destination address

IPv4 address of the S2L sub-LSP destination.

(There is NO-CONSENSUS amongst the authors on the sub-LSP ID described below and it needs more discussion)

Sub-LSP ID

A 16-bit identifier that identifies a particular instance

of a S2L sub-LSP. It can be varied for S2L sub-LSP make-before-break. Different S2L sub-LSPs, with the same SESSION object and LSP ID, follow the label merge semantics described in section 3 to form a particular instance of the P2MP tunnel.

20.3.2. S2L SUB-LSP IPv6 Object

SUB_LSP Class = TBD, S2L_SUB_LSP_IPv6 C-Type = TBD

This is same as the S2L IPv4 Sub-LSP object, with the difference that the destination address is a 16 byte IPv6 address.

20.4. FILTER_SPEC Object

The FILTER_SPEC object is canonical to the P2MP SENDER_TEMPLATE object.

20.4.1. P2MP LSP_TUNNEL_IPv4 FILTER_SPEC Object

Class = FILTER SPEC, P2MP LSP_TUNNEL_IPv4 C-Type = TBD

The format of the P2MP LSP_TUNNEL_IPv4 FILTER_SPEC object is identical to the P2MP LSP_TUNNEL_IPv4 SENDER_TEMPLATE object.

20.4.2. P2MP LSP_TUNNEL_IPv4 FILTER_SPEC Object

Class = FILTER SPEC, P2MP LSP_TUNNEL_IPv6 C_Type = TBD

The format of the P2MP LSP_TUNNEL_IPv6 FILTER_SPEC object is identical to the P2MP LSP_TUNNEL_IPv6 SENDER_TEMPLATE object.

20.5. SUB_EXPLICIT_ROUTE Object (SERO)

The SERO is defined as identical to the ERO. The CNums are TBD and TBD of the form 11bbbbbb.

20.6. SUB_RECORD_ROUTE Object (SRRO)

The SRRO is defined as identical to the RRO. The CNums are TBD and TBD of the form 11bbbbbb.

21. IANA Considerations

21.1. New Message Objects

IANA considerations for new message objects will be specified after the objects used are decided upon.

21.2. New Error Codes

Two new Error Codes are defined for use with the Error Value "Routing Error". IANA is requested to assign values.

The Error Code "Unable to Branch" indicates that a P2MP branch cannot be formed by the reporting LSR.

The Error Code "Unsupported LSP Integrity" indicates that a P2MP branch does not support the requested LSP integrity function.

21.3. LSP Attributes Flags

IANA has been asked to manage the space of flags in the Attibutes Flags TLV carried in the LSP_ATTRIBUTES Object [LSP-ATTRIB]. This document defines two new flags as follows:

```
Suggested Bit Number:            3
Meaning:                         LSP Integrity Required
Used in Attributes Flags on Path: Yes
Used in Attributes Flags on Resv: No
Used in Attributes Flags on RRO:  No
Referenced Section of this Document:   12

Suggested Bit Number:            4
Meaning:                         Branch Reoptimization Allowed
Used in Attributes Flags on Path: Yes
Used in Attributes Flags on Resv: No
Used in Attributes Flags on RRO:  No
Referenced Section of this Document: TBD
```

22. Security Considerations

This document does not introduce any new security issues. The security issues identified in [RFC3209] and [RFC3473] are still relevant.

23. Acknowledgements

This document is the product of many people. The contributors are listed in Section 25.

Thanks to Yakov Rekhter, Der-Hwa Gan, Arthi Ayyanger and Nischal Sheth for their suggestions and comments. Thanks also to Dino Farninacci for his comments.

24. Example P2MP LSP Establishment

Following is one example of setting up a P2MP LSP Tunnel using the procedures described in this document.

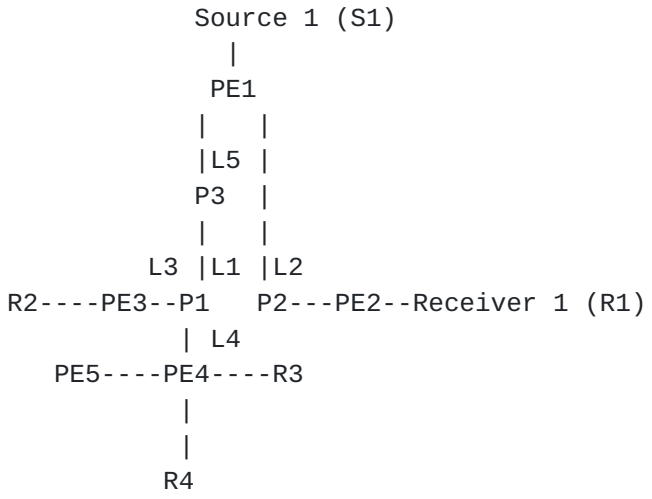


Figure 2.

The mechanism is explained using Figure 2. PE1 is the ingress-LSR. PE2, PE3 and PE4 are Egress-LSRs.

a) PE1 learns that PE2, PE3 and PE4 are interested in joining a P2MP tree with a P2MP ID of P2MP ID1. We assume that PE1 learns of the egress-LSRs at different points.

b) PE1 computes the P2P path to reach PE2.

c) PE1 establishes the S2L sub-LSP to PE2 along <PE1, P2, PE2>

d) PE1 computes the P2P path to reach PE3 when it discovers PE3. This path is computed to share the same links where possible with the sub-LSP to PE2 as they belong to the same P2MP session.

e) PE1 establishes the S2L sub-LSP to PE3 along <PE1, P3, P1, PE3>

f) PE1 computes the P2P path to reach PE4 when it discovers PE4. This path is computed to share the same links where possible with the sub-LSPs to PE2 and PE3 as they belong to the same P2MP session.

g) PE1 signals the Path message for PE4 sub-LSP along <PE1, P3, P1, PE4>

e) P1 receives a Resv message from PE4 with label L4. It had previously received a Resv message from PE3 with label L3. It had allocated a label L1 for the sub-LSP to PE3. It uses the same label and sends the Resv messages to P3. Note that it may send only one Resv message with multiple flow descriptors in the flow descriptor list. If this is the case and FF style is used, the FF flow descriptor will contain the S2L sub-LSP descriptor list with two entries: one for PE4 and the other for PE3. For SE style, the SE filter spec will contain this S2L sub-LSP descriptor list. P1 also creates a label mapping of (L1 -> {L3, L4}). P3 uses the existing label L5 and sends the Resv message to PE1, with label L5. It reuses the label mapping of {L5 -> L1}.

25. References

25.1. Normative References

[LSP-HIER] K. Kompella, Y. Rekhter, "LSP Hierarchy with Generalized MPLS TE", draft-ietf-mpls-lsp-hierarchy-08.txt.

[LSP-ATTR] A. Farrel, et. al. , "Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) Establishment Using RSVP-TE", draft-ietf-mpls-rsvpte-attributes-03.txt, March 2004, work in progress.

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

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

[RFC2205] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1, Functional Specification", RFC 2205, September 1997.

[RFC3471] Lou Berger, et al., "Generalized MPLS - Signaling Functional Description", RFC 3471, January 2003

[RFC3473] L. Berger et.al., "Generalized MPLS Signaling - RSVP-TE Extensions", RFC 3473, January 2003.

[RFC2961] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

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

[RSVP-FRR] P. Pan, G. Swallow, A. Atlas (Editors), "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", draft-ietf-mpls-rsvp-lsp-fastreroute-07.txt.

[P2MP-REQ] S. Yasukawa, et. al., "Requirements for Point-to-Multipoint capability extension to MPLS", draft-ietf-mpls-p2mp-sig-requirement-00.txt.

25.2. Informative References

[BFD] D. Katz, D. Ward, "Bidirectional Forwarding Detection", draft-katz-ward-bfd-01.txt.

[BFD-MPLS] R. Aggarwal, K. Kompella, "BFD for MPLS LSPs", draft-raggarwa-mpls-bfd-00.txt

[IPR-1] Bradner, S., "IETF Rights in Contributions", BCP 78, RFC 3667, February 2004.

[IPR-2] Bradner, S., Ed., "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3668, February 2004.

[INT-REG] JP Vasseur, A. Ayyangar, "Inter-area and Inter-AS MPLS Traffic Engineering", draft-vasseur-ccamp-inter-area-as-te-00.txt.

[RFC2209] R. Braden, L. Zhang, "Resource Reservation Protocol (RSVP) Version 1 Message Processing Rules", RFC 2209.

[RFC3477] K. Kompella, Y. Rekther, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)".

26. Author Information

26.1. Editor Information

Rahul Aggarwal
Juniper Networks
1194 North Mathilda Ave.
Sunnyvale, CA 94089
Email: rahul@juniper.net

Seisho Yasukawa
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 4769
EMail: yasukawa.seisho@lab.ntt.co.jp

Dimitri Papadimitriou
Alcatel
Francis Wellesplein 1,
B-2018 Antwerpen, Belgium
Phone: +32 3 240-8491
Email: Dimitri.Papadimitriou@alcatel.be

26.2. Contributor Information

John Drake
Calient Networks
Email: jdrake@calient.net

Alan Kullberg
Motorola Computer Group
120 Turnpike Road 1st Floor
Southborough, MA 01772
EMail: alan.kullberg@motorola.com

Lou Berger

Movaz Networks, Inc.
7926 Jones Branch Drive
Suite 615
McLean VA, 22102
Phone: +1 703 847-1801
EMail: lberger@movaz.com

Liming Wei
Redback Networks
350 Holger Way
San Jose, CA 95134
Email: lwei@redback.com

George Apostolopoulos
Redback Networks
350 Holger Way
San Jose, CA 95134
Email: georgeap@redback.com

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
Email: kireeti@juniper.net

George Swallow
Cisco Systems, Inc.
300 Beaver Brook Road
Boxborough , MA - 01719
USA
Email: swallow@cisco.com

JP Vasseur
Cisco Systems, Inc.
300 Beaver Brook Road
Boxborough , MA - 01719
USA
Email: jpv@cisco.com

Dean Cheng
Cisco Systems Inc.
170 W Tasman Dr.
San Jose, CA 95134
Phone 408 527 0677
Email: dcheng@cisco.com

Markus Jork
Avici Systems

101 Billerica Avenue
N. Billerica, MA 01862
Phone: +1 978 964 2142
EMail: mjork@avici.com

Hisashi Kojima
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 6070
EMail: kojima.hisashi@lab.ntt.co.jp

Andrew G. Malis
Tellabs
2730 Orchard Parkway
San Jose, CA 95134
Phone: +1 408 383 7223
Email: Andy.Malis@tellabs.com

Koji Sugisono
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 2605
EMail: sugisono.koji@lab.ntt.co.jp

Masanori Uga
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 4804
EMail: uga.masanori@lab.ntt.co.jp

Igor Bryskin
Movaz Networks, Inc.
7926 Jones Branch Drive
Suite 615
McLean VA, 22102

Adrian Farrel
Old Dog Consulting
Phone: +44 0 1978 860944
EMail: adrian@olddog.co.uk

Jean-Louis Le Roux
France Telecom
2, avenue Pierre-Marzin
22307 Lannion Cedex

France
E-mail: jeanlouis.leroux@francetelecom.com

27. Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

28. Full Copyright Statement

29. Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.