                            Extensions to
        Resource Reservation Protocol - Traffic Engineering (RSVP-TE)
            for Point-to-Multipoint TE Label Switched Paths (LSPs)

Abstract

   This document describes extensions to Resource Reservation Protocol -
   Traffic Engineering (RSVP-TE) for the set up 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.

Table of Contents

1. Introduction

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

    This document defines extensions to the RSVP-TE protocol ([RFC3209]
    and [RFC3473]) to support P2MP TE LSPs satisfying the set of
    requirements described in [RFC4461].

    This document relies on the semantics of the Resource Reservation
    Protocol (RSVP) that RSVP-TE inherits for building P2MP LSPs. A P2MP
    LSP is comprised of multiple source-to-leaf (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 for a single P2MP LSP. Hence the S2L sub-LSPs belonging
    to a P2MP LSP can be signaled using one Path message or split across
    multiple Path messages.

    There are various applications for P2MP TE LSPs and the signaling
    techniques described in this document can be used, sometimes in
    combination with other techniques, to support different applications.

    Specification of how applications will use P2MP TE LSPs and how the
    paths of P2MP TE LSPs are computed is outside the scope of this
    document.

2. 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 [RFC2119].

3. Terminology

    This document uses terminologies defined in [RFC2205], [RFC3031],
    [RFC3209], [RFC3473], [RFC4090], and [RFC4461].

4. 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 an LSR that replicates the incoming data on
   to one or more outgoing interfaces. The solution relies on RSVP-TE
   in the network for setting up a P2MP TE LSP.

   The P2MP TE LSP is set up by associating multiple S2L 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.

4.1. P2MP Tunnels

   The defining feature of a P2MP TE LSP is the action required at
   branch nodes where data replication occurs. Incoming MPLS labeled
   data is replicated to outgoing interfaces which may use different
   labels for the data.

   A P2MP TE Tunnel comprises one or more P2MP LSPs. A P2MP TE Tunnel
   is identified by a P2MP SESSION object. This object contains the
   identifier of the P2MP Session, which includes the P2MP Identifier
   (P2MP ID), a tunnel Identifier (Tunnel ID), and an extended tunnel
   identifier (Extended Tunnel ID). The P2MP ID is a four-octet number
   and is unique within the scope of the ingress LSR.

   The <P2MP ID, Tunnel ID, Extended Tunnel ID> tuple provides an
   identifier for the set of destinations of the P2MP TE Tunnel.

   The fields of the 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 ID field. The P2MP SESSION
   object is defined in section 19.1

4.2. P2MP LSP

   A P2MP LSP 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 19.2.

4.3. Sub-Groups

   As with all other RSVP controlled LSPs, P2MP LSP state is managed
   using RSVP messages. While the use of RSVP messages is the same,
   P2MP LSP state differs from P2P LSP state in a number of ways. A

P2MP LSP comprises multiple S2L Sub-LSPs, and as a result of this, it may not be possible to represent full state in 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 the P2MP LSP must also handle the state "re-merge" problem, see [RFC4461] and section 18.

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 the rest of the SENDER_TEMPLATE and SESSION objects, are used to represent a portion of a P2MP LSP's state. This portion of a P2MP LSP's state refers only to signaling state and not data plane replication or branching. For example, it is possible for a node to "branch" signaling state for a P2MP LSP, but to not branch the data associated with the P2MP LSP. Typical applications for generation and use of multiple sub-groups are (1) addition of an egress and (2) semantic fragmentation to ensure that a Path message remains within a single IP packet.

4.4. S2L Sub-LSPs

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

4.4.1. Representation of an S2L Sub-LSP

An S2L sub-LSP exists within the context of a P2MP LSP. 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 19.3.

An EXPLICIT_ROUTE Object (ERO) or P2MP_SECONDARY_EXPLICIT_ROUTE Object (SERO) is used to optionally specify the explicit route of a S2L sub-LSP. Each ERO or SERO that is signaled corresponds to a particular S2L_SUB_LSP object. Details of explicit route encoding are specified in section 4.5. The SECONDARY_EXPLICIT_ROUTE Object is defined in [RFC4873], a new P2MP SECONDARY_EXPLICIT_ROUTE Object C-type is defined in section 19.5, and a matching P2MP_SECONDARY_RECORD_ROUTE Object C-type is defined in section 19.6.

4.4.2. S2L Sub-LSPs and Path Messages

   The mechanism in this document allows a P2MP LSP 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 contain all the S2L sub-LSPs;
   and they also allow separate manipulation of sub-trees of the P2MP
   LSP. The reason for allowing a single Path message to signal
   multiple S2L sub-LSPs is to optimize the number of control messages
   needed to set up a P2MP LSP.

4.5. 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 encodes 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 and is referred to as the sub-LSP
   descriptor. 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.

   The path of each subsequent S2L sub-LSP is encoded in a
   P2MP_SECONDARY_EXPLICIT_ROUTE object (SERO). The format of the SERO
   is the same as an ERO (as defined in [RFC3209] and [RFC3473]). Each
   subsequent S2L sub-LSP is represented by tuples of the form < [<P2MP
   SECONDARY_EXPLICIT_ROUTE>], <S2L_SUB_LSP> >. An SERO for a
   particular S2L sub-LSP includes only the path from a branch LSR to
   the egress LSR of that S2L sub-LSP. The branch MUST appear as an
   explicit hop in the ERO or some other SERO. The absence of an 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 is
   described in detail in section 19.

   In order to avoid the potential repetition of path information for
   the parts of S2L sub-LSPs that share hops, this information is
   deduced from the explicit routes of other S2L sub-LSPs using explicit
   route compression in SEROs.

```
                        A
                        |
                        |
                        B
                        |
                        |
              C----D----E
              |    |    |
              |    |    |
              F    G    H-------I
                   |    |\      |
                   |    | \     |
                   |    |  \    |
                   J    K   L   M
                   |    |   |   |
                   |    |   |   |
                   N    O   P   Q--R
```
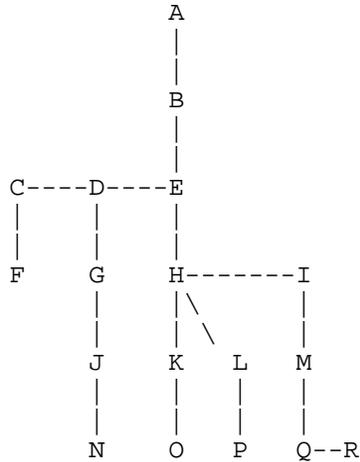
              Figure 1. Explicit Route Compression

Figure 1 shows a P2MP LSP with LSR A as the ingress LSR and six
egress LSRs: (F, N, O, P, Q and R). When all 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. The following encoding 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 the 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

   The following encoding 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 for the reader.

   This compression mechanism reduces the Path message size. It also
   reduces extra 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. An LSR needs
   to process an 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. The branch LSR that is the first hop of an SERO propagates
   the corresponding S2L sub-LSP downstream.

5. Path Message

5.1. 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>]
```

The 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>
                                [ <P2MP SECONDARY_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 SECONDARY-/EXPLICIT_ROUTE object
combination.

Per the definition of <S2L sub-LSP descriptor>, each S2L_SUB_LSP
object MAY be followed by a corresponding SERO. The first
S2L_SUB_LSP object is a special case, and its explicit route is
specified by the ERO. Therefore, the first S2L_SUB_LSP object SHOULD
NOT be followed by an SERO, and if one is present, it MUST be
ignored.

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

An IF_ID RSVP_HOP object MUST be used on links where there is not a
one-to-one association of a control channel to a data channel
[RFC3471]. An RSVP_HOP object defined in [RFC2205] SHOULD be used
otherwise.

Path message processing is described in the next section.

5.2. Path Message Processing

   The ingress LSR initiates the setup of an S2L sub-LSP to each egress
   LSR that is a destination of the P2MP LSP. Each S2L sub-LSP is
   associated with the same P2MP LSP using common P2MP SESSION object
   and <Sender Address, LSP-ID> fields in the P2MP SENDER_TEMPLATE
   object. Hence, it can be combined with other S2L sub-LSPs to form a
   P2MP LSP. Another S2L sub-LSP belonging to the same instance of this
   S2L sub-LSP (i.e., the same P2MP LSP) SHOULD share resources with
   this S2L sub-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 in one or more Path messages, and a given Path message
   can contain one or more S2L sub-LSPs. An LSR that supports RSVP-TE
   signaled P2MP LSPs MUST be able to receive and process multiple Path
   messages for the same P2MP LSP and multiple S2L sub-LSPs in one Path
   message. This implies that such an LSR MUST be able to receive and
   process all objects listed in section 19.

5.2.1. Multiple Path Messages

   As described in section 4, either the < [<EXPLICIT_ROUTE>]
   <S2L_SUB_LSP> > or the < [<P2MP SECONDARY_EXPLICIT_ROUTE>]
   <S2L_SUB_LSP> > tuple is used to specify an S2L sub-LSP. Multiple
   Path messages can be used to signal a P2MP LSP. 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 5.2.2.

   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. Or it may be that when new leaves are
   added to the P2MP LSP, they are signaled in a new Path message. Or
   an ingress LSR MAY choose to break the P2MP tree into separate
   manageable P2MP 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 (the P2MP
   Tree) and multiple trees with a single leaf each (S2L sub-LSPs). Per
   [RFC4461], a P2MP LSP MUST have consistent attributes across all
   portions of a tree. This implies that each Path message that is used
   to signal a P2MP LSP is signaled using the same signaling attributes

   with the exception of the S2L sub-LSP descriptors and Sub-Group
   identifier.

   The resulting sub-LSPs from the different Path messages belonging to
   the same P2MP LSP 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. In this case, the message can be decomposed
   into multiple Path messages such that each message carries a subset
   of the X2L sub-tree carried by the incoming message.

   Multiple Path messages generated by an 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 fields) and encoded in the SENDER_TEMPLATE object. Multiple
   Path messages generated by an 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 MUST be set to the TE Router
   ID of the LSR that originates the Path message. 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 is globally
   unique. The Sub-Group ID space is specific to the Sub-Group
   Originator ID.

5.2.2. 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. Each S2L sub-LSP descriptor
   describes a single S2L sub-LSP.

   All LSRs MUST process the ERO corresponding to the first S2L sub-LSP
   if the ERO is present. If one or more SEROs are present, an ERO MUST
   be present. The first S2L sub-LSP MUST be propagated in a Path
   message by each LSR along the explicit route specified by the ERO, if
   the ERO is present. Else it MUST be propagated using hop-by-hop
   routing towards the destination identified by the S2L_SUB_LSP object.

   An LSR MUST process an S2L sub-LSP descriptor for a subsequent S2L
   sub-LSP as follows:

   If the S2L_SUB_LSP object is followed by an SERO, the LSR MUST check
   the first hop in the SERO:

- If the first hop of the SERO identifies a local address of the
  LSR, and the LSR is also the egress identified by the
  S2L_SUB_LSP object, the descriptor MUST NOT be propagated
  downstream, but the SERO may be used for egress control per
  [RFC4003].

- If the first hop of the SERO identifies a local address of the
  LSR, and the LSR is not the egress as identified by the
  S2L_SUB_LSP object, the S2L sub-LSP descriptor MUST be included
  in a Path message sent to the next-hop determined from the SERO.

- If the first hop of the SERO is not a local address of the LSR,
  the S2L sub-LSP descriptor MUST be included in the Path message
  sent to the 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 the S2L_SUB_LSP object is not followed by an SERO, the LSR MUST
examine the S2L_SUB_LSP object:

- If this LSR is the egress as identified by the S2L_SUB_LSP
  object, the S2L sub-LSP descriptor MUST NOT be propagated
  downstream.

- If this LSR is not the egress as identified by the S2L_SUB_LSP
  object, the LSR MUST make a routing decision to determine the
  next hop towards the egress, and MUST include the S2L sub-LSP
  descriptor in a Path message sent to the next-hop towards the
  egress. In this case, the LSR MAY insert an SERO into the S2L
  sub-LSP descriptor.

Hence, a branch LSR MUST only propagate the relevant S2L sub-LSP
descriptors to each downstream hop. An S2L sub-LSP descriptor list
that is propagated on a downstream link MUST only contain those S2L
sub-LSPs that are routed using that hop. This processing MAY result
in a subsequent S2L sub-LSP in an incoming Path message becoming the
first S2L sub-LSP in an outgoing Path message.

Note that if one or more SEROs contain loose hops, expansion of such
loose hops MAY result in overflowing the Path message size. section
5.2.3 describes how signaling of the set of S2L sub-LSPs can be split
across 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 MUST append its address in an incoming RRO
and propagate it downstream. A branch LSR MUST form a new RRO for
each of the outgoing Path messages by copying the RRO from the

incoming Path message and appending its address. Each such updated
RRO MUST be formed using the rules in [RFC3209] (and updated by
[RFC3473]), as appropriate.

If an LSR is unable to support an S2L sub-LSP in a Path message (for
example, it is unable to route towards the destination using the
SERO), a PathErr message MUST be sent for the impacted S2L sub-LSP,
and normal processing of the rest of the P2MP LSP 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_Removed flag MUST NOT be set. However, the ingress LSR
may set an LSP Integrity flag to request that if there is a setup
failure on any branch, the entire LSP should fail to set up. This is
described further in sections 5.2.4 and 11.

5.2.3. Transit Fragmentation of Path State Information

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. RSVP [RFC2205] disallows the use of IP
fragmentation, and thus IP fragmentation MUST be avoided in this
case. In order to achieve this, the multiple Path messages generated
by the transit LSR are signaled with the Sub-Group Originator ID set
to the TE Router ID of the transit LSR and with a distinct Sub-Group
ID for each Path message. Thus, each distinct Path message that is
generated by the transit LSR for the P2MP LSP 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 descriptor (e.g., SERO), message and hop information
(e.g., INTEGRITY, MESSAGE_ID, and RSVP_HOP), and the Sub-Group fields
of the SENDER_TEMPLATE objects. Except when a make-before-break
operation is being performed (as specified in section 14.1), the
tunnel sender address and LSP ID fields MUST be the same in each
message. For transit nodes, they MUST be the same as the values in
the received Path message.

As described above, one case in which the Sub-Group Originator ID of
a received Path message is changed is that of fragmentation of a Path
message at a transit node. Another case is when the Sub-Group
Originator ID of a received Path message may 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, an 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 changed.

5.2.4. 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 to fail.

The ingress LSP may request 'LSP integrity' by setting bit 3 of the
Attributes Flags TLV. The bit is set if LSP integrity is required.

It is RECOMMENDED to use the LSP_REQUIRED_ATTRIBUTES object
[RFC4420].

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 message carrying the error "Routing Error"/"Unsupported LSP
Integrity".

5.3. Grafting

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

There are two methods to add S2L sub-LSPs to a P2MP LSP. The first
is to add new S2L sub-LSPs to the P2MP LSP 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. 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 10.1.
The egress LSRs can be added 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.

6. Resv Message

6.1. 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 flow descriptor list>]

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

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

   <S2L sub-LSP flow descriptor> ::= <S2L_SUB_LSP>
                                    [<P2MP_SECONDARY_RECORD_ROUTE>]

   FILTER_SPEC is defined in section 19.4.

The S2L sub-LSP flow descriptor has the same format as S2L sub-LSP
descriptor in section 4.1 with the difference that a
P2MP_SECONDARY_RECORD_ROUTE object is used in place of a P2MP
SECONDARY_EXPLICIT_ROUTE object. The P2MP_SECONDARY_RECORD_ROUTE
objects follow the same compression mechanism as the P2MP
SECONDARY_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 SHOULD be
allocated if the <Sender Address, LSP-ID> fields of the FILTER_SPEC
object are the same.

However different labels MUST be allocated if the <Sender Address,
LSP-ID> of the FILTER_SPEC object is different, as that implies that
the FILTER_SPEC refers to a different P2MP LSP.

6.2. Resv Message Processing

The egress LSR MUST follow normal RSVP procedures while originating a
Resv message. The format of Resv messages is as defined in section
6.1. As usual, the Resv message carries the label allocated by the
egress LSR.

A node upstream of the egress node MUST allocate its own label and
pass 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 MUST
NOT be sent upstream until at least one Resv message has been
received from a downstream neighbor. When the integrity bit is set
in the LSP_REQUIRED_ATTRIBUTE object, Resv message MUST NOT be sent
upstream until all Resv messages have been received from the
downstream neighbors.

Each Fixed-Filter (FF) flow descriptor or Shared-Explicit (SE) filter
spec sent upstream in a Resv message includes an S2L sub-LSP
descriptor list. Each such FF flow descriptor or SE filter spec for
the same P2MP LSP (whether on one or multiple Resv messages) on the
same Resv MUST be allocated the same label, and FF flow descriptors
or SE filter specs SHOULD use the same label across multiple Resv
messages.

The node that sends the Resv message, for a P2MP LSP, upstream MUST
associate the label assigned by this node with all the labels
received from downstream Resv messages, for that P2MP LSP. 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 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 MAY forward a single Resv message upstream for each
received Resv message from a downstream receiver. Note that there
may be a large number of Resv messages at and close to the ingress
LSR for an LSP with many receivers. A branch LSR SHOULD combine Resv
state from multiple receivers into a single Resv message to be sent
upstream (see section 6.2.1). However, note that this may result in
overflowing the Resv message, particularly as the number of receivers
downstream of any branch LSR increases as the LSR is closer to the
ingress LSR. Thus, a branch LSR MAY choose to send more than one
Resv message upstream and partition the Resv state between the
messages.

When a transit node sets the Sub-Group Originator field in a Path
message, it MUST replace the Sub-Group fields received in the
FILTER_SPEC objects of any associated Resv messages with the value
that it originally received in the Sub-Group fields of the Path
message from the upstream neighbor.

ResvErr message generation is unmodified. Nodes propagating a
received ResvErr message MUST use the Sub-Group field values carried
in the corresponding Resv message.

6.2.1. Resv Message Throttling

A branch node may have to send a revised Resv message upstream
whenever there is a change in a Resv message for an S2L sub-LSP
received from one of the downstream neighbors. This can result in
excessive Resv messages sent upstream, particularly when the S2L sub-
LSPs are first established. In order to mitigate this situation,
branch nodes can 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 P2MP_SECONDARY_RECORD_ROUTE objects
(SRROs), 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. A suggested value for the delay time
is thirty seconds, and delay times SHOULD generally be longer than 1
second. Specific mechanisms for Resv message throttling and delay
timer settings are implementation dependent and are outside the scope
of this document.

6.3. Route Recording

6.3.1. RRO Processing

   A Resv message for a P2P LSP contains a recorded route if the ingress
   LSR requested route recording by including an RRO in the original
   Path message. The same rule is used during signaling of P2MP LSPs.
   That is, inclusion of an RRO in the Path message used to signal one
   or more S2L sub-LSPs triggers the inclusion of a recorded route for
   each sub-LSP in the Resv message.

   The recorded route of the first S2L sub-LSP is encoded in the RRO.
   Additional recorded routes for the subsequent S2L sub-LSPs are
   encoded in P2MP_SECONDARY_RECORD_ROUTE objects (SRROs). Their format
   is specified in section 19.5. Each S2L_SUB_LSP object in a Resv is
   associated with an RRO or SRRO. The first S2L_SUB_LSP object (for
   the first S2L sub-LSP) is associated with the RRO. Subsequent
   S2L_SUB_LSP objects (for subsequent S2L sub-LSPs) are each followed
   by an SRRO that contains the recorded route for that S2L sub-LSP from
   the leaf to a branch. The ingress node can then use the RRO and
   SRROs to determine the end-to-end path for each S2L sub-LSP.

6.4. Reservation Style

   Considerations about the reservation style in a Resv message apply as
   described in [RFC3209]. The reservation style in the Resv messages
   can be either FF or SE. All P2MP LSPs that belong to the same P2MP
   Tunnel MUST be signaled with the same reservation style.
   Irrespective of whether the reservation style is FF or SE, the S2L
   sub-LSPs that belong to the same P2MP LSP SHOULD share labels where
   they share hops. If the S2L sub-LSPs that belong to the same P2MP
   LSP share labels then they MUST share resources. If the reservation
   style is FF, then S2L sub-LSPs that belong to different P2MP LSPs
   MUST NOT share resources or labels. If the reservation style is SE,
   then S2L sub-LSPs that belong to different P2MP LSPs and the same
   P2MP tunnel SHOULD share resources where they share hops, but they
   MUST not share labels in packet environments.

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>]

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

   The definition of <sender descriptor> is not changed by this
   document.

7.2. Pruning

   The operation of removing egress LSR(s) from an existing P2MP LSP is
   termed as pruning. This operation allows egress nodes to be removed
   from a P2MP LSP at different points in time. This section describes
   the mechanisms to perform pruning.

7.2.1. Implicit S2L Sub-LSP Teardown

   Implicit teardown uses standard RSVP message processing. Per
   standard RSVP processing, an 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 an 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.

   When implicit teardown is used to delete one or more S2L sub-LSPs, by
   modifying a Path message, a transit LSR may have to generate a
   PathTear message downstream to delete one or more of these S2L sub-
   LSPs. This can happen if as a result of the implicit deletion of S2L
   sub-LSP(s) there are no remaining S2L sub-LSPs to send in the
   corresponding Path message downstream.

7.2.2. Explicit S2L Sub-LSP Teardown

   Explicit S2L Sub-LSP teardown relies on generating a PathTear message
   for the corresponding Path message. The PathTear message is signaled
   with the SESSION and SENDER_TEMPLATE objects corresponding to the
   P2MP LSP and the <Sub-Group Originator ID, Sub-Group ID> tuple
   corresponding to the Path message. This approach SHOULD be used when
   all the egresses signaled by a Path message need to be removed from
   the P2MP LSP. Other S2L sub-LSPs, from other sub-groups signaled
   using other Path messages, are not affected by the PathTear.

   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 in the Path message that
   was used to set up the S2L sub-LSPs being torn down. 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.

   When a P2MP LSP is removed by the ingress, a PathTear message MUST be
   generated for each Path message used to signal the P2MP LSP.

8. Notify and ResvConf Messages

8.1. Notify Messages

   The Notify Request object and Notify message are described in
   [RFC3473]. Both object and message SHALL be supported for delivery
   of upstream and downstream notification. Processing not detailed in
   this section MUST comply to [RFC3473].

   1. Upstream Notification

   If a transit LSR sets the Sub-Group Originator ID in the
   SENDER_TEMPLATE object of a Path message to its own address, and the
   incoming Path message carries a Notify Request object, then this LSR
   MUST change the Notify node address in the Notify Request object to
   its own address in the Path message that it sends.

   If this LSR subsequently receives a corresponding Notify message from
   a downstream LSR, then it MUST:

      - send a Notify message upstream toward the Notify node address
        that the LSR received in the Path message.

       - process the Sub-Group fields of the SENDER_TEMPLATE object on
         the received Notify message, and modify their values, in the
         Notify message that is forwarded, to match the Sub-Group field
         values in the original Path message received from upstream.

   The receiver of an (upstream) Notify message MUST identify the state
   referenced in this message based on the SESSION and SENDER_TEMPLATE.

   2. Downstream Notification

   A transit LSR sets the Sub-Group Originator ID in the FILTER_SPEC
   object(s) of a Resv message to the value that was received in the
   corresponding Path message. If the incoming Resv message carries a
   Notify Request object, then:

       - If there is at least another incoming Resv message that carries
         a Notify Request object, and the LSR merges these Resv messages
         into a single Resv message that is sent upstream, the LSR MUST
         set the Notify node address in the Notify Request object to its
         Router ID.

       - Else if the LSR sets the Sub-Group Originator ID (in the
         outgoing Path message that corresponds to the received Resv
         message) to its own address, the LSR MUST set the Notify node
         address in the Notify Request object to its Router ID.

       - Else the LSR MUST propagate the Notify Request object unchanged,
         in the Resv message that it sends upstream.

   If this LSR subsequently receives a corresponding Notify message from
   an upstream LSR, then it MUST:

       - process the Sub-Group fields of the FILTER_SPEC object in the
         received Notify message, and modify their values, in the Notify
         message that is forwarded, to match the Sub-Group field values
         in the original Path message sent downstream by this LSR.

       - send a Notify message downstream toward the Notify node address
         that the LSR received in the Resv message.

   The receiver of a (downstream) Notify message MUST identify the state
   referenced in the message based on the SESSION and FILTER_SPEC
   objects.

   The consequence of these rules for a P2MP LSP is that an upstream
   Notify message generated on a branch will result in a Notify being
   delivered to the upstream Notify node address. The receiver of the
   Notify message MUST NOT assume that the Notify message applies to all

downstream egresses, but MUST examine the information in the message
to determine to which egresses the message applies.

Downstream Notify messages MUST be replicated at branch LSRs
according to the Notify Request objects received on Resv messages.
Some downstream branches might not request Notify messages, but all
that have requested Notify messages MUST receive them.

8.2. ResvConf Messages

ResvConf messages are described in [RFC2205]. ResvConf processing in
[RFC3473] and [RFC3209] is taken directly from [RFC2205]. An egress
LSR MAY include a RESV_CONFIRM object that contains the egress LSR's
address. The object and message SHALL be supported for the
confirmation of receipt of the Resv message in P2MP TE LSPs.
Processing not detailed in this section MUST comply to [RFC2205].

A transit LSR sets the Sub-Group Originator ID in the FILTER_SPEC
object(s) of a Resv message to the value that was received in the
corresponding Path message. If any of the incoming Resv messages
corresponding to a single Path message carry a RESV_CONFIRM object,
then the LSR MUST include a RESV_CONFIRM object in the corresponding
Resv message that it sends upstream. If the Sub-Group Originator ID
is its own address, then it MUST set the receiver address in the
RESV_CONFIRM object to this address, else it MUST propagate the
object unchanged.

A transit LSR sets the Sub-Group Originator ID in the FILTER_SPEC
object(s) of a Resv message to the value that was received in the
corresponding Path message. If an incoming Resv message
corresponding to a single Path message carries a RESV_CONFIRM object,
then the LSR MUST include a RESV_CONFIRM object in the corresponding
Resv message that it sends upstream and:

   - If there is at least another incoming Resv message that carries
     a RESV_CONFIRM object, and the LSR merges these Resv messages
     into a single Resv message that is sent upstream, the LSR MUST
     set the receiver address in the RESV_CONFIRM object to its
     Router ID.

   - If the LSR sets the Sub-Group Originator ID (in the outgoing
     Path message that corresponds to the received Resv message) to
     its own address, the LSR MUST set the receiver address in the
     RESV_CONFIRM object to its Router ID.

   - Else the LSR MUST propagate the RESV_CONFIRM object unchanged,
     in the Resv message that it sends upstream.

   If this LSR subsequently receives a corresponding ResvConf message
   from an upstream LSR, then it MUST:

      - process the Sub-Group fields of the FILTER_SPEC object in the
        received ResvConf message, and modify their values, in the
        ResvConf message that is forwarded, to match the Sub-Group field
        values in the original Path message sent downstream by this LSR.

      - send a ResvConf message downstream toward the receiver address
        that the LSR received in the RESV_CONFIRM object in the Resv
        message.

   The receiver of a ResvConf message MUST identify the state referenced
   in this message based on the SESSION and FILTER_SPEC objects.

   The consequence of these rules for a P2MP LSP is that a ResvConf
   message generated at the ingress will result in a ResvConf message
   being delivered to the branch and then to the receiver address in the
   original RESV_CONFIRM object. The receiver of a ResvConf message
   MUST NOT assume that the ResvConf message should be sent to all
   downstream egresses, but it MUST replicate the message according to
   the RESV_CONFIRM objects received in Resv messages. Some downstream
   branches might not request ResvConf messages, and ResvConf messages
   SHOULD NOT be sent on these branches. All downstream branches that
   requested ResvConf messages MUST be sent such a message.

9. Refresh Reduction

   The refresh reduction procedures described in [RFC2961] are equally
   applicable to P2MP LSPs 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 LSPs.

10. State Management

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

   Additional information signaled in the Path/Resv message is part of
   the state created by a local implementation. This includes PHOP/NHOP
   and SENDER_TSPEC/FILTER_SPEC objects.

10.1. 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/from a P2MP LSP
   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 to or deleted from the
   P2MP LSP by allowing a Path or a PathTear message to incrementally
   change the existing P2MP LSP Path state.

   As described in section 5.2, multiple Path messages can be used to
   signal a P2MP LSP. 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 state updates.

10.2. Combining Multiple Path Messages

   There is a tradeoff between the number of Path messages used by the
   ingress to maintain the P2MP LSP and the processing imposed by full
   state messages when adding S2L sub-LSPs to an existing Path message.
   It is possible to combine S2L sub-LSPs previously advertised in
   different Path messages in 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
   that 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 MAY contain new Sub-Group ID field values. 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 (see section 3.1.3 of [RFC2205]). At that point the S2L sub-LSP SHOULD be deleted from the old Path state using the procedures of section 7.

A Path message with a Sub-Group_ID(n) may signal a set of S2L sub-LSPs that belong partially or entirely to an already existing Sub-Group_ID(i), or a strictly non-overlapping new set of S2L sub-LSPs. A newly received Path message that matches SESSION object and Sender Tunnel Address, LSP ID, Sub-Group Originator ID> with existing Path state carrying the same or different Sub-Group_ID, referred to Sub-Group_ID(n) is processed as follows:

1) If Sub-Group_ID(i) = Sub-Group_ID(n), then S2L Sub-LSPs that are in both Sub-Group_ID(i) and Sub-Group_ID(n) are refreshed. New S2L Sub-LSPs are added to Sub-Group_ID(i) Path state and S2L Sub-LSPs that are in Sub-Group_ID(i) but not in Sub-Group_ID(n) are deleted from the Sub-Group_ID(i) Path state.

2) If Sub-Group_ID(i) != Sub-Group_ID(n), then a new Sub-Group_ID(n) Path state is created for S2L Sub-LSPs signaled by Sub-Group_ID(n). S2L Sub-LSPs in existing Sub-Group_IDs(i) Path state (that are or are not in the newly received Path message Sub-Group_ID(n)) are left unmodified (see above).

11. Error Processing

   PathErr and ResvErr messages are processed as per RSVP-TE procedures. Note that an 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. If the ingress node requests 'LSP integrity', an error reported on a branch of a P2MP TE LSP for a particular S2L sub-LSP may change the state of any other S2L sub-LSP of the same P2MP TE LSP. This is explained further in section 11.3.

11.1. PathErr Messages

   The PathErr message will include one or more S2L_SUB_LSP objects.
   The resulting modified format for 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 message 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 PathErr message. The S2L sub-LSP descriptor list is
   defined in section 5.1.

11.2. ResvErr Messages

   The ResvErr message will include one or more S2L_SUB_LSP objects.
   The resulting modified format for a ResvErr Message is:

   <ResvErr Message> ::=   <Common Header> [<INTEGRITY>]
                             [[<MESSAGE_ID_ACK> |
                                <MESSAGE_ID_NACK>] ...]
                             [<MESSAGE_ID>]
                             <SESSION> <RSVP_HOP>
                             <ERROR_SPEC> [<SCOPE>]
                             [<ACCEPTABLE_LABEL_SET> ...]
                             [<POLICY_DATA> ...]
                             <STYLE> <flow descriptor list>

   ResvErr message generation is unmodified, but nodes that set the
   Sub-Group Originator field and propagate a received ResvErr message
   downstream MUST replace the Sub-Group fields received in the ResvErr
   message with the value that was set in the Sub-Group fields of the
   Path message sent to the downstream neighbor. Note the receiver of a
   ResvErr message is able to identify the errored outgoing Resv

message, and outgoing interface, based on the Sub-Group fields
received in the ResvErr message. The flow descriptor list is defined
in section 6.1.

11.3. Branch Failure Handling

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 message 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_REQUIRED_ATTRIBUTEs
object in section 5.2.4), 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 SHOULD set the Path_State_Removed flag.

A branch LSR that receives a PathErr message during LSP setup with
the Path_State_Removed flag set MUST act according to the wishes of
the ingress LSR. The default behavior is that the branch LSR clears
the Path_State_Removed flag on the PathErr and sends it further
upstream. It does not tear any other branches of the LSP. However,
if the LSP integrity flag is set on the Path message, the branch LSR
MUST send PathTear on all other downstream branches and send the
PathErr message upstream with the Path_State_Removed flag set.

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 reported by received PathErr messages and all
branches that are explicitly torn by the branch LSR.

12. 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 message) on all
   branches before relaying a corresponding Resv message upstream.

13. Label Allocation on LANs with Multiple Downstream Nodes

   A branch LSR of a P2MP LSP on an Ethernet LAN segment SHOULD send one
   copy of the data traffic per downstream LSR connected on that LAN for
   that P2MP LSP. Procedures for preventing MPLS labeled traffic
   replication in such a case is beyond the scope of this document.

14. P2MP LSP and Sub-LSP Re-Optimization

   It is possible to change the path used by P2MP LSPs to reach the
   destinations of the P2MP tunnel. There are two methods that can be
   used to accomplish this. The first is make-before-break, defined in
   [RFC3209], and the second uses the sub-groups defined above.

14.1. Make-before-Break

   In this case, all the S2L sub-LSPs are signaled with a different LSP
   ID by the ingress LSR and follow the make-before-break procedure
   defined in [RFC3209]. Thus, a new P2MP LSP is established. Each S2L
   sub-LSP is signaled with a different LSP ID, corresponding to the new
   P2MP LSP. After moving traffic to the new P2MP LSP, the ingress can
   tear down the old P2MP LSP. This procedure can be used to re-
   optimize the path of the entire P2MP LSP or the paths to a subset of
   the destinations of the P2MP LSP. When modifying just a portion of
   the P2MP LSP, this approach requires the entire P2MP LSP to be re-
   signaled.

14.2. Sub-Group-Based Re-Optimization

   Any node may initiate re-optimization of a set of S2L sub-LSPs by
   using incremental state update and then, optionally, combining
   multiple path messages.

To alter the path taken by a particular set of S2L sub-LSPs, the node
initiating the path change initiates one or more separate Path
messages for the same P2MP LSP, each with a new sub-Group ID. The
generation of these Path messages, each with one or more S2L sub-
LSPs, follows procedures in section 5.2. As is the case in section
10.2, a particular egress continues to be advertised in both the old
and new Path messages until a Resv message listing the egress and
corresponding to the new Path message is received by the re-
optimizing node. At that point, the egress SHOULD be deleted from
the old Path state using the procedures of section 7. Sub-tree re-
optimization is then completed.

Sub-Group-based re-optimization may result in transient data
duplication as the new Path messages for a set of S2L sub-LSPs may
transit one or more nodes with the old Path state for the same set of
S2L sub-LSPs.

As is always the case, a node may choose to combine multiple path
messages as described in section 10.2.

15. Fast Reroute

[RFC4090] extensions can be used to perform fast reroute for the
mechanism described in this document when applied within packet
networks. GMPLS introduces other protection techniques that can be
applied to packet and non-packet environments [RFC4873], but which
are not discussed further in this document. This section only
applies to LSRs that support [RFC4090].

This section uses terminology defined in [RFC4090], and fast reroute
procedures defined in [RFC4090] MUST be followed unless specified
below. The head-end and transit LSRs MUST follow the
SESSION_ATTRIBUTE and FAST_REROUTE object processing as specified in
[RFC4090] for each Path message and S2L sub-LSP of a P2MP LSP. Each
S2L sub-LSP of a P2MP LSP MUST have the same protection
characteristics. The RRO processing MUST apply to SRRO as well
unless modified below.

The sections that follow describe how fast reroute may be applied to
P2MP MPLS TE LSPs in all of the principal operational scenarios.
This document does not describe the detailed processing steps for
every imaginable usage case, and they may be described in future
documents, as needed.

15.1. Facility Backup

   Facility backup can be used for link or node protection of LSRs on
   the path of a P2MP LSP. The downstream labels MUST be learned by the
   Point of Local Repair (PLR), as specified in [RFC4090], from the
   label corresponding to the S2L sub-LSP in the RESV message.
   Processing of SEROs signaled in a backup tunnel MUST follow backup
   tunnel ERO processing described in [RFC4090].

15.1.1. Link Protection

   If link protection is desired, a bypass tunnel MUST be used to
   protect the link between the PLR and next-hop. Thus all S2L sub-LSPs
   that use the link SHOULD be protected in the event of link failure.
   Note that all such S2L sub-LSPs belonging to a particular instance of
   a P2MP tunnel SHOULD share the same outgoing label on the link
   between the PLR and the next-hop as per section 5.2.1. This is the
   P2MP LSP label on the link. Label stacking is used to send data for
   each P2MP LSP into the bypass tunnel. The inner label is the P2MP
   LSP label allocated by the next-hop.

   During failure, Path messages for each S2L sub-LSP that is affected,
   MUST be sent to the Merge Point (MP) by the PLR. It is RECOMMENDED
   that the PLR uses 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 MUST use the LSP-ID to identify the corresponding S2L sub-
   LSPs. The MP MUST NOT use the <Sub-Group Originator ID, Sub-Group
   ID> tuple while identifying the corresponding S2L sub-LSPs. In order
   to further process an S2L sub-LSP the MP MUST determine the protected
   S2L sub-LSP using the LSP-ID and the S2L_SUB_LSP object.

15.1.2. Node Protection

   If node protection is desired the PLR SHOULD use one or more P2P
   bypass tunnels to protect the set of S2L sub-LSPs that transit the
   protected node. Each of these P2P bypass tunnels MUST intersect the
   path of the S2L sub-LSPs that they protect on an LSR that is
   downstream from the protected node. This constrains the set of S2L
   sub-LSPs being backed- up via that bypass tunnel to those S2L sub-
   LSPs that pass through a common downstream MP. This MP is the
   destination of the bypass tunnel. When the PLR forwards incoming
   data for a P2MP LSP into the bypass tunnel, the outer label is the
   bypass tunnel label and the inner label is the label allocated by the
   MP to the set of S2L sub-LSPs belonging to that P2MP LSP.

After detecting failure of the protected node the PLR MUST send one
or more Path messages for all protected S2L sub-LSPs to the MP of the
protected S2L sub-LSP. 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 MUST use the LSP-ID to identify the corresponding
S2L sub-LSPs. The MP MUST NOT use the <Sub-Group Originator ID,
Sub-Group ID> tuple while identifying the corresponding S2L sub-LSPs
because the Sub-Group Originator ID might be changed by some LSR that
is bypassed by the bypass tunnel. In order to further process an S2L
sub-LSP the MP MUST determine the protected S2L sub-LSP using the
LSP-ID and the S2L_SUB_LSP object.

Note that node protection MAY require the PLR to be branch capable in
the data plane, as multiple bypass tunnels may be required to back up
the set of S2L sub-LSPs passing through the protected node. If the
PLR is not branch capable, the node protection mechanism described
here is applicable to only those cases where all the S2L sub-LSPs
passing through the protected node also pass through a single MP that
is downstream from the protected node. A PLR MUST set the Node
protection flag in the RRO/SRRO as specified in [RFC4090]. If a PLR
is not branch capable, and one or more S2L sub-LSPs are added to a
P2MP tree, and these S2L sub-LSPs do not transit the existing MP
downstream of the protected node, then the PLR MUST reset this flag.

It is to be noted that procedures in this section require P2P bypass
tunnels. Procedures for using P2MP bypass tunnels are for further
study.

15.2. One-to-One Backup

One-to-one backup, as described in [RFC4090], 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 SHOULD
share the same P2MP LSP label, as per section 5.2.1. All such S2L
sub-LSPs belonging to a P2MP LSP MUST be protected.

The backup S2L sub-LSPs may traverse different next-hops at the PLR.
Thus, the set of outgoing labels and next-hops for a P2MP LSP, at the
PLR, may change once protection is triggered. Consider a P2MP LSP
that is using a single next-hop and label between the PLR and the
next-hop of the PLR. This may no longer be the case once protection
is triggered. This MAY require a PLR to be branch capable in the
data plane. If the PLR is not branch capable, the one-to-one backup
mechanisms described here are only applicable to those cases where
all the backup S2L sub-LSPs pass through the same next-hop downstream

of the PLR. Procedures for one-to-one backup when a PLR is not
branch capable and when all the backup S2L sub-LSPs do not pass
through the same downstream next-hop are for further study.

It is recommended that the path-specific method be used to identify a
backup S2L sub-LSP. Hence, the DETOUR object SHOULD 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 4, S2L sub-LSPs between different P2MP tunnel instances use
different labels.

If there is only one 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 object applies to all the S2L sub-
LSPs.

16. 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 Code "Routing Error" and Error Value
"Unable to Branch".

It is 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
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 [LSP-STITCH] and
LSP hierarchy [RFC4206]. 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 [RFC4206] allows P2MP LSPs
to be built in such an environment. A P2P LSP segment is signaled
from the last P2MP-capable hop that is upstream of a legacy LSR to
the first P2MP-capable hop that is downstream of it. This assumes
that intermediate legacy LSRs are transit LSRs: they cannot act as
P2MP branch points. Transit LSRs along this LSP segment do not
process control plane messages associated with the P2MP LSP.
Furthermore, these transit LSRs also do not need to have P2MP data

plane capabilities as they only need to process data belonging to the
P2P LSP segment. Hence, these transit LSRs do not need to support
P2MP MPLS. This P2P LSP segment is stitched to the incoming P2MP
LSP. 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.

In packet networks, the S2L sub-LSPs may be nested inside the outer
P2P LSP. Hence, label stacking can be used to enable use of the same
LSP segment for multiple P2MP LSPs. Stitching and nesting
considerations and procedures are described further in [LSP-STITCH]
and [RFC4206].

There maybe 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 P2MP-capable LSRs [TE-NODE-CAP]. It can use
this information to compute S2L sub-LSP paths such that they avoid
legacy non-P2MP-capable LSRs. The explicit route object of an 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 up to 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 Traffic
Engineering Database (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] or [BFD-MPLS]) to the next P2MP-
capable LSR. This may be needed for the directed Path message head-
end to use node protection fast reroute 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.

17. Reduction in Control Plane Processing with LSP Hierarchy

It is possible to take advantage of LSP hierarchy [RFC4206] while
setting up P2MP LSP, 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, do not process control plane messages associated with the P2MP
LSP. In fact, 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 LSPs can be set up dynamically as described in the
previous section or preconfigured. For example, in Figure 2 in
section 24, PE1 can set up 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 and does not
process the P2MP control messages.

18. P2MP LSP Re-Merging and Cross-Over

   This section details the procedures for detecting and dealing with
   re-merge and cross-over. The term "re-merge" refers to the case of
   an ingress or transit node that creates a branch of a P2MP LSP, a re-
   merge branch, that intersects the P2MP LSP at another node farther
   down the tree. This may occur due to such events as an error in path
   calculation, an error in manual configuration, or network topology
   changes during the establishment of the P2MP LSP. If the procedures
   detailed in this section are not followed, data duplication will
   result.

   The term "cross-over" refers to the case of an ingress or transit
   node that creates a branch of a P2MP LSP, a cross-over branch, that
   intersects the P2MP LSP at another node farther down the tree. It is
   unlike re-merge in that, at the intersecting node, the cross-over
   branch has a different outgoing interface as well as a different
   incoming interface. This may be necessary in certain combinations of
   topology and technology; e.g., in a transparent optical network in
   which different wavelengths are required to reach different leaf
   nodes.

   Normally, a P2MP LSP has a single incoming interface on which all of
   the data for the P2MP LSP is received. The incoming interface is
   identified by the IF_ID RSVP_HOP object, if present, and by the
   interface over which the Path message was received if the IF_ID
   RSVP_HOP object is not present. However, in the case of dynamic LSP
   re-routing, the incoming interface may change.

   Similarly, in both the re-merge and cross-over cases, a node will
   receive a Path message for a given P2MP LSP identifying a different
   incoming interface for the data, and the node needs to be able to
   distinguish between dynamic LSP re-routing and the re- merge/cross-
   over cases.

Make-before-break represents yet another similar but different case,
in that the incoming interface associated with the make-before-break
P2MP LSP may be different than that associated with the original P2MP
LSP. However, the two P2MP LSPs will be treated as distinct (but
related) LSPs because they will have different LSP ID field values in
their SENDER_TEMPLATE objects.

18.1. Procedures

When a node receives a Path message, it MUST check whether it has
matching state for the P2MP LSP. Matching state is identified by
comparing the SESSION and SENDER_TEMPLATE objects in the received
Path message with the SESSION and SENDER_TEMPLATE objects of each
locally maintained P2MP LSP Path state. The P2MP ID, Tunnel ID, and
Extended Tunnel ID in the SESSION object and the sender address and
LSP ID in the SENDER_TEMPLATE object are used for the comparison. If
the node has matching state, and the incoming interface for the
received Path message is different than the incoming interface of the
matching P2MP LSP Path state, then the node MUST determine whether it
is dealing with dynamic LSP rerouting or re-merge/cross-over.

Dynamic LSP rerouting is identified by checking whether there is any
intersection between the set of S2L_SUB_LSP objects associated with
the matching P2MP LSP Path state and the set of S2L_SUB_LSP objects
in the received Path message. If there is any intersection, then
dynamic re-routing has occurred. If there is no intersection between
the two sets of S2L_SUB_LSP objects, then either re-merge or cross-
over has occurred. (Note that in the case of dynamic LSP rerouting,
Path messages for the non-intersecting members of set of S2L_SUB_LSPs
associated with the matching P2MP LSP Path state will be received
subsequently on the new incoming interface.)

In order to identify the re-merge case, the node processing the
received Path message MUST identify the outgoing interfaces
associated with the matching P2MP Path state. Re-merge has occurred
if there is any intersection between the set of outgoing interfaces
associated with the matching P2MP LSP Path state and the set of
outgoing interfaces in the received Path message.

18.1.1. Re-Merge Procedures

There are two approaches to dealing with the re-merge case. In the
first, the node detecting the re-merge case, i.e., the re-merge node,
allows the re-merge case to persist, but data from all but one
incoming interface is dropped at the re-merge node. In the second,
the re-merge node initiates the removal of the re-merge branch(es)
via signaling. Which approach is used is a matter of local policy.

A node MUST support both approaches and MUST allow user configuration
of which approach is to be used.

When configured to allow a re-merge case to persist, the re-merge
node MUST validate consistency between the objects included in the
received Path message and the matching P2MP LSP Path state. Any
inconsistencies MUST result in a PathErr message sent to the previous
hop of the received Path message. The Error Code is set to "Routing
Problem", and the Error Value is set to "P2MP Re-Merge Parameter
Mismatch".

If there are no inconsistencies, the node logically merges, from the
downstream perspective, the control state of incoming Path message
with the matching P2MP LSP Path state. Specifically, procedures
related to processing of messages received from upstream MUST NOT be
modified from the upstream perspective; this includes processing
related to refresh and state timeout. In addition to the standard
upstream related procedures, the node MUST ensure that each object
received from upstream is appropriately represented within the set of
Path messages sent downstream. For example, the received <S2L sub-
LSP descriptor list> MUST be included in the set of outgoing Path
messages. If there are any NOTIFY_REQUEST objects present, then the
procedures defined in section 8 MUST be followed for all Path and
Resv messages. Special processing is also required for Resv
processing. Specifically, any Resv message received from downstream
MUST be mapped into an outgoing Resv message that is sent to the
previous hop of the received Path message. In practice, this
translates to decomposing the complete <S2L sub-LSP descriptor list>
into subsets that match the incoming Path messages, and then
constructing an outgoing Resv message for each incoming Path message.

When configured to allow a re-merge case to persist, the re-merge
node receives data associated with the P2MP LSP on multiple incoming
interfaces, but it MUST only send the data from one of these
interfaces to its outgoing interfaces. That is, the node MUST drop
data from all but one incoming interface. This ensures that
duplicate data is not sent on any outgoing interface. The mechanism
used to select the incoming interface is implementation specific and
is outside the scope of this document.

When configured to correct the re-merge branch via signaling, the re-
merge node MUST send a PathErr message corresponding to the received
Path message. The PathErr message MUST include all of the objects
normally included in a PathErr message, as well as one or more
S2L_SUB_LSP objects from the set of sub-LSPs associated with the
matching P2MP LSP Path state. A minimum of three S2L_SUB_LSP objects
is RECOMMENDED. This will allow the node that caused the re-merge to
identify the outgoing Path state associated with the valid portion of

the P2MP LSP. The set of S2L_SUB_LSP objects in the received Path
message MUST also be included. The PathErr message MUST include the
Error Code "Routing Problem" and Error Value of "P2MP Re-Merge
Detected". The node MAY set the Path_State_Removed flag [RFC3473].
As is always the case, the PathErr message is sent to the previous
hop of the received Path message.

A node that receives a PathErr message that contains the Error Value
"Routing Problem/P2MP Re-Merge Detected" MUST determine if it is the
node that created the re-merge case. This is done by checking
whether there is any intersection between the set of S2L_SUB_LSP
objects associated with the matching P2MP LSP Path state and the set
of other-branch S2L_SUB_LSP objects in the received PathErr message.
If there is, then the node created the re-merge case. Other-branch
S2L_SUB_LSP objects are those S2L_SUB_LSP objects included, by the
node detecting the re-merge case, in the PathErr message that were
taken from the matching P2MP LSP Path state. Such S2L_SUB_LSP
objects are identifiable as they will not be included in the Path
message associated with the received PathErr message. See section
11.1 for more details on how such an association is identified.

The node SHOULD remove the re-merge case by moving the S2L_SUB_LSP
objects included in the Path message associated with the received
PathErr message to the outgoing interface associated with the
matching P2MP LSP Path state. A trigger Path message for the moved
S2L_SUB_LSP objects is then sent via that outgoing interface. If the
received PathErr message did not have the Path_State_Removed flag
set, the node SHOULD send a PathTear via the outgoing interface
associated with the re-merge branch.

If use of a new outgoing interface violates one or more SERO
constraints, then a PathErr message containing the associated
egresses and any identified S2L_SUB_LSP objects SHOULD be generated
with the Error Code "Routing Problem" and Error Value of "ERO
Resulted in Re-Merge".

The only case where this process will fail is when all the listed
S2L_SUB_LSP objects 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.

19. New and Updated Message Objects

   This section presents the RSVP object formats as modified by this
   document.

19.1. SESSION Object

   A P2MP LSP 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
   that are part of the same P2MP LSP 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 LSPs,
   each distinguished by a unique SENDER_TEMPLATE object.

19.1.1. P2MP LSP Tunnel IPv4 SESSION Object

   Class = SESSION, P2MP_LSP_TUNNEL_IPv4 C-Type = 13

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            P2MP ID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      MUST be zero              |            Tunnel ID          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Extended Tunnel ID                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
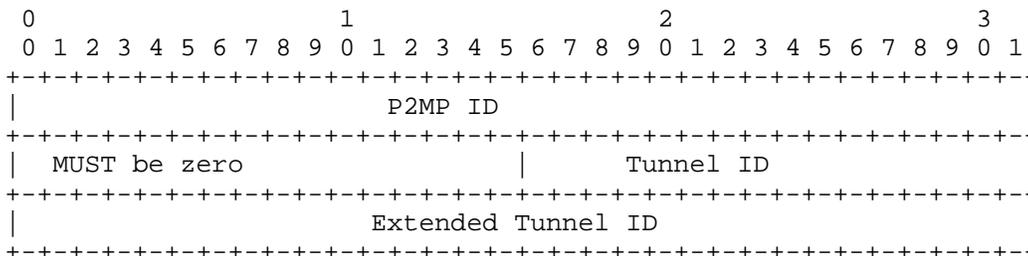
   P2MP ID
      A 32-bit identifier used in the SESSION object that remains
      constant over the life of the P2MP tunnel. It encodes the P2MP
      Identifier that is unique within the scope of the ingress LSR.

   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. Ingress LSRs that wish
       to have a globally unique identifier for the P2MP tunnel SHOULD
       place their tunnel sender address here. A combination of this
       address, P2MP ID, and Tunnel ID provides a globally unique
       identifier for the P2MP tunnel.

19.1.2. P2MP LSP Tunnel IPv6 SESSION Object

   This is the 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].

   Class = SESSION, P2MP_LSP_TUNNEL_IPv6 C-Type = 14

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          P2MP ID                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        MUST be zero            |           Tunnel ID           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Extended Tunnel ID (16 bytes)               |
   |                                                               |
   |                          .......                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
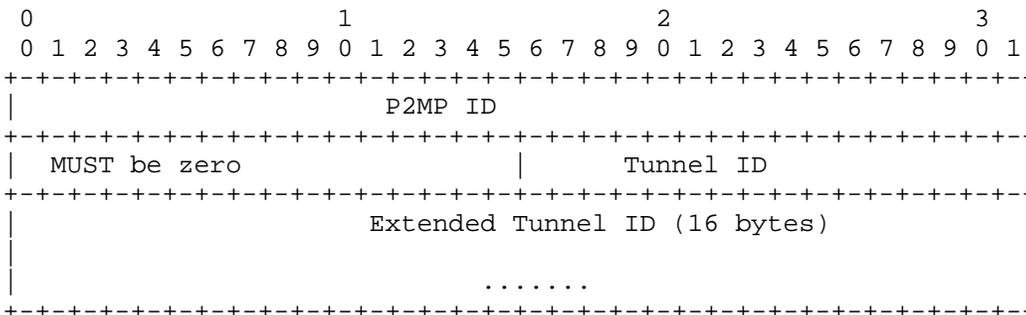
19.2. SENDER_TEMPLATE Object

   The SENDER_TEMPLATE object contains the ingress LSR source address.
   The 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. 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 by
   using a <Sub-Group ID Originator ID, Sub-Group ID> tuple. The
   SENDER_TEMPLATE object is modified to carry this information as shown
   below.

19.2.1. P2MP LSP Tunnel IPv4 SENDER_TEMPLATE Object

   Class = SENDER_TEMPLATE, P2MP_LSP_TUNNEL_IPv4 C-Type = 12
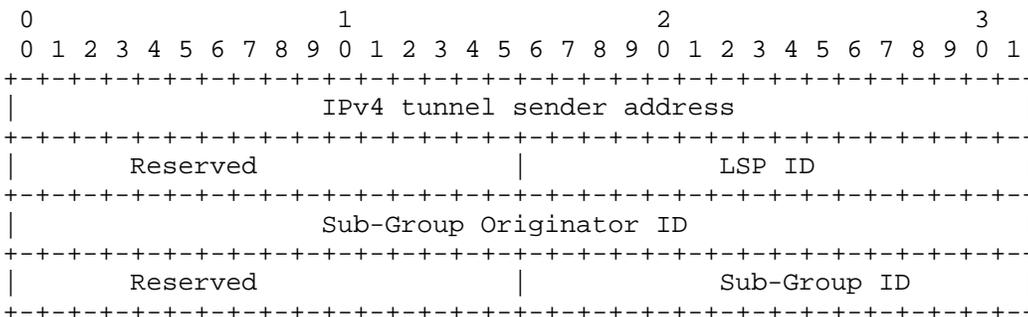
```
         0                   1                   2                   3
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                    IPv4 tunnel sender address                 |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |        Reserved               |             LSP ID            |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                    Sub-Group Originator ID                    |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |        Reserved               |          Sub-Group ID         |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   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 an 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.

   LSP ID
      See [RFC3209].

19.2.2. P2MP LSP Tunnel IPv6 SENDER_TEMPLATE Object

   Class = SENDER_TEMPLATE, P2MP_LSP_TUNNEL_IPv6 C-Type = 13
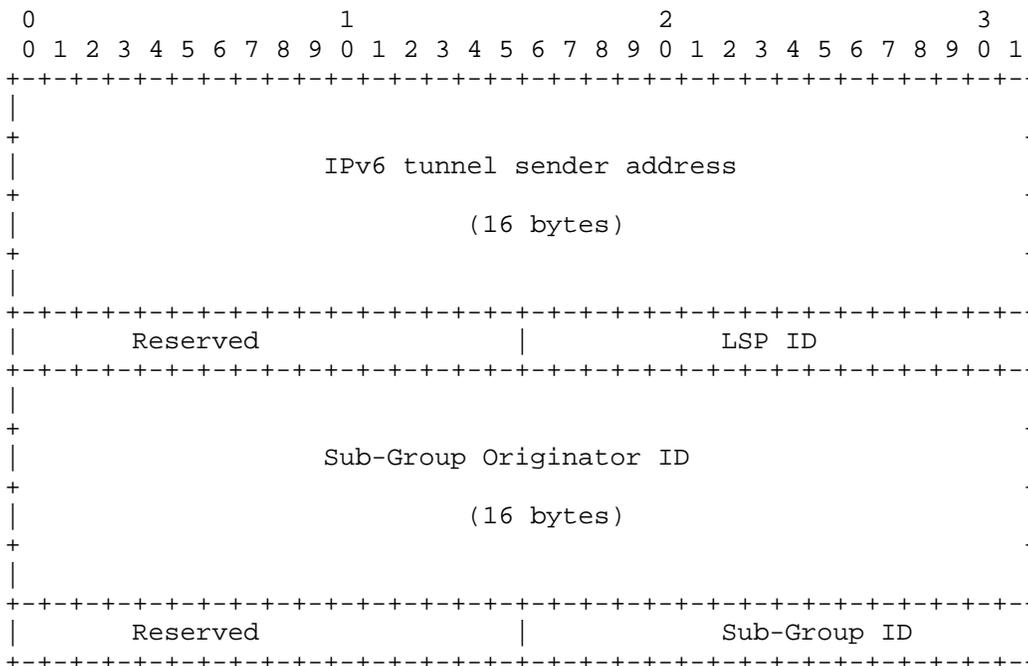
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                   IPv6 tunnel sender address                  |
   +                                                               +
   |                          (16 bytes)                           |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Reserved            |            LSP ID             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                   Sub-Group Originator ID                     |
   +                                                               +
   |                          (16 bytes)                           |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Reserved            |          Sub-Group ID        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   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 an LSR which re-originates the Path message with its own
      Sub-Group Originator ID.

   Sub-Group ID
      As above in section 19.2.1.

   LSP ID
      See [RFC3209].

19.3. S2L_SUB_LSP Object

   An S2L_SUB_LSP object identifies a particular S2L sub-LSP belonging
   to the P2MP LSP.

19.3.1. S2L_SUB_LSP IPv4 Object

   S2L_SUB_LSP Class = 50, S2L_SUB_LSP_IPv4 C-Type = 1

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 IPv4 S2L Sub-LSP destination address          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
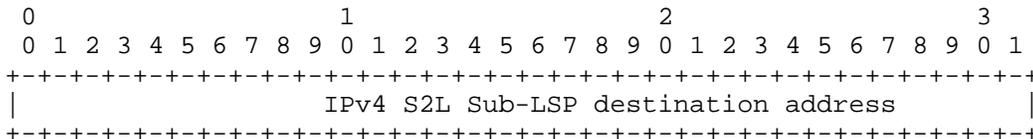
   IPv4 Sub-LSP destination address
      IPv4 address of the S2L sub-LSP destination.

19.3.2. S2L_SUB_LSP IPv6 Object

   S2L_SUB_LSP Class = 50, S2L_SUB_LSP_IPv6 C-Type = 2

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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          IPv6 S2L Sub-LSP destination address (16 bytes)       |
   |                            ....                                |
   |                                                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
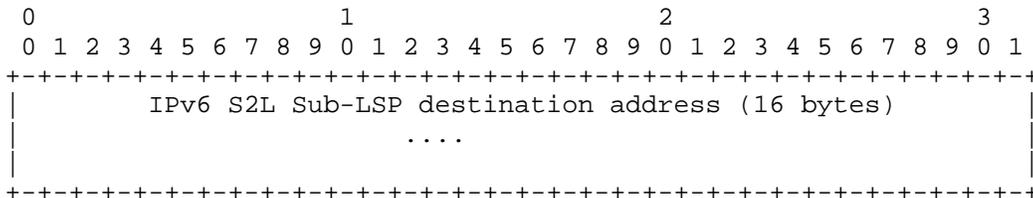
19.4. FILTER_SPEC Object

   The FILTER_SPEC object is canonical to the P2MP SENDER_TEMPLATE
   object.

19.4.1. P2MP LSP_IPv4 FILTER_SPEC Object

   Class = FILTER_SPEC, P2MP LSP_IPv4 C-Type = 12

   The format of the P2MP LSP_IPv4 FILTER_SPEC object is identical to
   the P2MP LSP_IPv4 SENDER_TEMPLATE object.

19.4.2. P2MP LSP_IPv6 FILTER_SPEC Object

   Class = FILTER_SPEC, P2MP LSP_IPv6 C-Type = 13

   The format of the P2MP LSP_IPv6 FILTER_SPEC object is identical to
   the P2MP LSP_IPv6 SENDER_TEMPLATE object.

19.5. P2MP SECONDARY_EXPLICIT_ROUTE Object (SERO)

   The P2MP SECONDARY_EXPLICIT_ROUTE Object (SERO) is defined as
   identical to the ERO. The class of the P2MP SERO is the same as the
   SERO defined in [RFC4873]. The P2MP SERO uses a new C-Type = 2. The
   sub-objects are identical to those defined for the ERO.

19.6. P2MP SECONDARY_RECORD_ROUTE Object (SRRO)

   The P2MP SECONDARY_RECORD_ROUTE Object (SRRO) is defined as identical
   to the ERO. The class of the P2MP SRRO is the same as the SRRO
   defined in [RFC4873]. The P2MP SRRO uses a new C-Type = 2. The
   sub-objects are identical to those defined for the RRO.

20. IANA Considerations

20.1. New Class Numbers

   IANA has assigned the following Class Numbers for the new object
   classes introduced. The Class Types for each of them are to be
   assigned via standards action. The sub-object types for the P2MP
   SECONDARY_EXPLICIT_ROUTE and P2MP_SECONDARY_RECORD_ROUTE follow the
   same IANA considerations as those of the ERO and RRO [RFC3209].

   50 Class Name = S2L_SUB_LSP

   C-Type
      1   S2L_SUB_LSP_IPv4 C-Type
      2   S2L_SUB_LSP_IPv6 C-Type

20.2. New Class Types

   IANA has assigned the following C-Type values:

   Class Name = SESSION

   C-Type
     13   P2MP_LSP_TUNNEL_IPv4 C-Type
     14   P2MP_LSP_TUNNEL_IPv6 C-Type

    Class Name = SENDER_TEMPLATE

    C-Type
      12   P2MP_LSP_TUNNEL_IPv4 C-Type
      13   P2MP_LSP_TUNNEL_IPv6 C-Type

    Class Name = FILTER_SPEC

    C-Type
      12   P2MP LSP_IPv4 C-Type
      13   P2MP LSP_IPv6 C-Type

    Class Name = SECONDARY_EXPLICIT_ROUTE (Defined in [RFC4873])

    C-Type
       2 P2MP SECONDARY_EXPLICIT_ROUTE C-Type

    Class Name = SECONDARY_RECORD_ROUTE (Defined in [RFC4873])

    C-Type
       2 P2MP_SECONDARY_RECORD_ROUTE C-Type

20.3. New Error Values

    Five new Error Values are defined for use with the Error Code
    "Routing Problem". IANA has assigned values for them as follows.

    The Error Value "Unable to Branch" indicates that a P2MP branch
    cannot be formed by the reporting LSR. IANA has assigned value 23 to
    this Error Value.

    The Error Value "Unsupported LSP Integrity" indicates that a P2MP
    branch does not support the requested LSP integrity function. IANA
    has assigned value 24 to this Error Value.

    The Error Value "P2MP Re-Merge Detected" indicates that a node has
    detected re-merge. IANA has assigned value 25 to this Error Value.

    The Error Value "P2MP Re-Merge Parameter Mismatch" is described in
    section 18. IANA has assigned value 26 to this Error Value.

    The Error Value "ERO Resulted in Re-Merge" is described in section
    18. IANA has assigned value 27 to this Error Value.

20.4. LSP Attributes Flags

   IANA has been asked to manage the space of flags in the Attributes
   Flags TLV carried in the LSP_REQUIRED_ATTRIBUTES object [RFC4420].
   This document defines a new flag as follows:

   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 Doc: 5.2.4

21. Security Considerations

   In principle this document does not introduce any new security issues
   above those identified in [RFC3209], [RFC3473], and [RFC4206].
   [RFC2205] specifies the message integrity mechanisms for hop-by-hop
   RSVP signaling. These mechanisms apply to the hop-by-hop P2MP RSVP-
   TE signaling in this document. Further, [RFC3473] and [RFC4206]
   specify the security mechanisms for non hop-by-hop RSVP-TE signaling.
   These mechanisms apply to the non hop-by-hop P2MP RSVP-TE signaling
   specified in this document, particularly in sections 16 and 17.

   An administration may wish to limit the domain over which P2MP TE
   tunnels can be established. This can be accomplished by setting
   filters on various ports to deny action on a RSVP path message with a
   SESSION object of type P2MP_LSP_IPv4 or P2MP_LSP_IPv6.

   The ingress LSR of a P2MP TE LSP determines the leaves of the P2MP TE
   LSP based on the application of the P2MP TE LSP. The specification
   of how such applications will use a P2MP TE LSP is outside the scope
   of this document. Applications MUST provide a mechanism to notify
   the ingress LSR of the appropriate leaves for the P2MP LSP.
   Specifications of applications within the IETF MUST specify this
   mechanism in sufficient detail that an ingress LSR from one vendor
   can be used with an application implementation provided by another
   vendor. Manual configuration of security parameters when other
   parameters are auto-discovered is generally not sufficient to meet
   security and interoperability requirements of IETF specifications.

22. Acknowledgements

   This document is the product of many people. The contributors are
   listed in Appendix B.

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

23. References

23.1. Normative References

   [RFC4206]   Kompella, K. and Y. Rekhter, "Label Switched Paths
                  (LSP) Hierarchy with Generalized Multi-Protocol Label
                  Switching (GMPLS) Traffic Engineering (TE)", RFC 4206,
                  October 2005.

   [RFC4420]   Farrel, A., Ed., Papadimitriou, D., Vasseur, J.-P., and
                  A. Ayyangar, "Encoding of Attributes for Multiprotocol
                  Label Switching (MPLS) Label Switched Path (LSP)
                  Establishment Using Resource ReserVation Protocol-
                  Traffic Engineering (RSVP-TE)", RFC 4420, February
                  2006.

   [RFC3209]   Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan,
                  V., and G. Swallow, "RSVP-TE: Extensions to RSVP for
                  LSP Tunnels", RFC 3209, December 2001.

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

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

   [RFC3471]   Berger, L., Ed., "Generalized Multi-Protocol Label
                  Switching (GMPLS) Signaling Functional Description",
                  RFC 3471, January 2003.

   [RFC3473]   Berger, L., Ed., "Generalized Multi-Protocol Label
                  Switching (GMPLS) Signaling Resource ReserVation
                  Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC
                  3473, January 2003.

   [RFC2961]   Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F.,
                  and 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.

   [RFC4090]   Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed.,
                  "Fast Reroute Extensions to RSVP-TE for LSP Tunnels",
                  RFC 4090, May 2005.

   [RFC3477]   Kompella, K. and Y. Rekhter, "Signalling Unnumbered
                  Links in Resource ReSerVation Protocol - Traffic
                  Engineering (RSVP-TE)", RFC 3477, January 2003.

   [RFC4873]   Berger, L., Bryskin, I., Papadimitriou, D., and A.
                  Farrel, "GMPLS Segment Recovery", RFC 4873, April 2007.

23.2. Informative References

   [RFC4461]   Yasukawa, S., Ed., "Signaling Requirements for Point-
                  to-Multipoint Traffic-Engineered MPLS Label Switched
                  Paths (LSPs)", RFC 4461, April 2006.

   [BFD]   Katz, D. and D. Ward, "Bidirectional Forwarding
                  Detection", Work in Progress, March 2007.

   [BFD-MPLS]   Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow,
                  "BFD for MPLS LSPs", Work in Progress, March 2007.

   [LSP-STITCH]   Ayyanger, A., Kompella, K., Vasseur, JP., and A.
                  Farrel, "Label Switched Path Stitching with Generalized
                  Multiprotocol Label Switching Traffic Engineering
                  (GMPLS TE)", Work in Progress, March 2007.

   [TE-NODE-CAP] Vasseur, JP., Ed., Le Roux, JL., Ed., "IGP Routing
                  Protocol Extensions for Discovery of Traffic
                  Engineering Node Capabilities", Work in Progress, April
                  2007.

   [RFC4003]   Berger, L., "GMPLS Signaling Procedure for Egress
                  Control", RFC 4003, February 2005.

Appendix A.   Example of P2MP LSP Setup

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

```
                  Source 1 (S1)
                    |
                   PE1
                  |    |
                  |L5  |
                  P3   |
                  |    |
             L3  |L1  |L2
        R2----PE3--P1    P2---PE2--Receiver 1 (R1)
                |  L4
           PE5----PE4----R3
                  |
                  |
                 R4
```
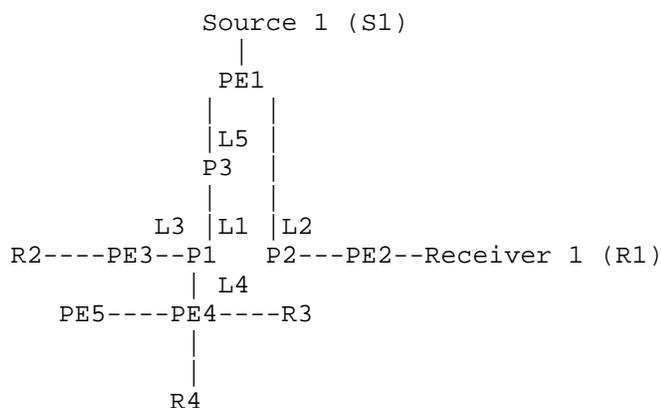
                   Figure 2.

   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 in time.

   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>.

   h) 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}.

Appendix B. Contributors

   John Drake
   Boeing
   EMail: john.E.Drake2@boeing.com

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

   Lou Berger
   LabN Consulting, L.L.C.
   EMail: lberger@labn.net

   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
   ibryskin@movaz.com
   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
   EMail: jeanlouis.leroux@francetelecom.com

Editors' Addresses

   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-lucent.be