Network Working Group   Seisho Yasukawa (NTT) - Editor
Internet Draft   Alan Kullberg (Motorola) - Editor
Expiration Date: August 2004   Lou Berger (Movaz) - Editor

February 2004

Extended RSVP-TE for Point-to-Multipoint LSP Tunnels

draft-yasukawa-mpls-rsvp-p2mp-04.txt

Status of this Memo

Abstract

This document describes a solution for point-to-multipoint (P2MP) Traffic Engineering (TE) which extends "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, and "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, to support P2MP TE LSPs. A P2MP TE LSP is established by setting up multiple standard point-to-point (P2P) TE LSPs from a sender node and all the downstream branch nodes along the P2P TE LSP to one of the leaf nodes of the P2MP TE LSP. A branched LSP is associated with original trunk LSP by newly defined Association object so that the P2MP LSP tunnel is established over the P2MP path. Because only a single standard LSP will be present on any given link along the P2MP LSP, the defined approach realizes maximum compatibility with existing implementations. The solution supports standard tree operations: setup, graft/join, and prune/leave. As the (G)MPLS signaling is used, the P2MP TE LSP can be built using any switching technology supported by GMPLS. This includes non-packet technologies.

Contents

1. Introduction

Point-to-multipoint (P2MP) technology will become increasingly important with the dissemination of new, real-time applications, such as content delivery services and video conferences, which require P2MP real-time transmission capability with much more bandwidth and stricter QoS than non-real-time applications.

This document defines RSVP-TE [RFC3209] and [RFC3473]protocol extensions in order to establish, maintain, and teardown a P2MP TE label switched path(LSP) [P2MP-REQ].

The use of label switching routers (LSRs) with these extensions allows service providers to offer services that utilize point-to-point (P2P) and/or P2MP multiprotocol label switching (MPLS), and Generalized MPLS (GMPLS), in the same service network.

These RSVP-TE protocol extensions are very flexible and can be used to carry protocols other than IP multicasting, e.g., Ethernet, PPP, and SONET/SDH. No assumption or restrictions are made about the format of the data to be carried in the signaled LSP.

2. Definitions

2.1. Terminology

The reader is assumed to be familiar with the terminology in [RFC3209], [RFC2205], [RFC3031], [RFC3473] and [P2MP-REQ].

trunk path: A main P2P path which composes the P2MP path and starts from an ingress/branch LSR of the P2MP path and ends with a downstream egress LSR. This path is encoded as an ERO by ingress LSR.

branch path:A branch P2P path which composes the P2MP path and starts from a branch LSR on the trunk LSP and ends with a egress LSR. This path is encoded as an SERO by ingress LSR.

2.2. Conventions

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

3. Problem statements

3.1. Motivation

This document provides a P2MP TE solution which satisfies requirements described in [P2MP-REQ]. The proposed solution defines the protocol mechanisms and signalling procedures for P2MP TE LSP. It does not define any mechanisms or procedures related to application specific requirements described in [P2MP-REQ] and these are outside the scope of this document.

3.2. Technical Objectives

The technical objectives described in this section is to meet all of the requirements set out in [P2MP-REQ].

```
                                  S1
                                  |
                                I-LSR1
                                  |
                                  |
                          +----- LSR2-----+
                          |               |
                          |               |
                        LSR3             LSR4(P2P)
                       +    +             |
                       |    |             |
                 R1---LSR5  LSR6(P2P)    LSR7---E-LSR8---R5
                       |    |             :
                       |    |             :
                    E-LSR9  LSR10(P2P)    :
                       |    |             :
                       |    |            E-LSR11
                       R2  E-LSR12(P2P)   :
                            |             :
                            |             R4
                            R3
```

Figure 1. P2MP TE LSP and source/receivers

The figure above shows a single ingress LSR (I-LSR1), and five egress LSRs (LSR5, E-LSR9, E-LSR12, E-LSR11 and E-LSR8). I-LSR1 accommodates a traffic source that is generating traffic for a P2MP application. Receivers:R1, R2, R3, R4 and R5 are attached to LSR5, E-LSR9, E-LSR12 E-LSR11 and E-LSR8 respectively.

The following are the technical objectives that we wish to achieve:

a) A P2MP TE path which satisfies various constraints is pre-determined and supplied to ingress I-LSR1 or dynamically calculated by P2MP path calculation engine located on the ingress I-LSR1.

   Typical constraints are bandwidth requirements, resource class affinities, fast rerouting, preemption, as in (G)MPLS. This document introduces a new constraint that branch nodes must be P2MP capable.

b) Ingress I-LSR1 sets up a P2MP TE LSP by means of P2MP signalling procedures and mechanism defined in this document from I-LSR1 to E-LSR9, E-LSR12, E-LSR11 and E-LSR8.

c) In this case, I-LSR1 associates a LABEL with incoming data traffic and tunnels this traffic into an established P2MP TE LSP based on FEC. Then branch LSR2 and LSR3 replicate incoming data traffic and sends the replicated traffic to multiple downstream LSRs, using the appropriate label to each LSR. Note that this P2MP label swapping relation is already established on the branch nodes by the above signalling procedures. Finally LSR5, E-LSR9, E-LSR12, E-LSR11 and E-LSR8 terminates the LSP and transmits the data traffic to each receiver.

d) If I-LSR1 decides to transmit P2MP data to E-LSR11 which accommodates a receiver (R4) who expresses an interest in receiving data, a new path is determined and a sub-P2MP path from LSR7 to E-LSR11 is grafted onto the P2MP path. Vice versa, if I-LSR1 decides to stop transmitting the data to E-LSR11, a sub-P2MP path from LSR7 to E-LSR11 is pruned from the P2MP path.

e) Note that legacy (P2P capable only) LSRs (LSR4, LSR6, LSR10 and E-LSR12) exist on the established P2MP TE LSP. These LSRs can serve as P2P transit (LSR4, LSR6 and LSR10) and egress LSR (E-LSR12) of P2MP TE LSP.

f) Note also that LSR5 serves as both egress (for R1) and transit (for E-LSR9) LSR of P2MP TE LSP.

4. Architecture

4.1. P2MP LSP tunnels

This solution defines a "P2MP flow" by a label that is used with the data traffic associated with a P2MP TE LSP. Such a P2MP TE LSP is referred to as a "P2MP LSP tunnel" because the traffic through it is opaque to intermediate nodes along the LSP.

This solution establishes a P2MP LSP tunnel by dividing it into multiple P2P LSP tunnels. The P2P LSP tunnels are concatenated to provide the service required by the P2MP LSP tunnel. Each constituent P2P LSP tunnel may be supported by multiple P2P TE LSPs.

A sample topology that can be supported using the defined P2MP approach is shown in Figure 2. In this example there is a single ingress LSR of the P2MP LSP tunnel, node A. There are multiple egress LSRs, nodes E, G, I, L, M, and N. There are multiple branch nodes, nodes B, D, F, K, M. Note that node M is both a egress and a branch node. There are also transit nodes, nodes C, H and J.

```
                (Ingress)  (Transit )  (Egress)
                           (& Branch)

                                      H----I
                                      |
                                      F----G
                                      |
                       A----B----C----D----E
                            |
                            |         +----L
                            |        /
                            J-------K------M
                                           |
                                           N
```

Figure 2: Sample P2MP LSP tunnel topology

With the defined solution (and assuming no load sharing or protection function), only a single LSP will be present on any given link along the P2MP path. As viewed at a single link in the network, the LSP will use standard LSP semantics to represent one path through the network. This path will have a specific starting point, which will either be the ingress or a branch node. The path will have a specific endpoint, which will be a single egress node. The endpoints will be indicated using standard unicast addresses, in standard SENDER_TEMPLATE and SESSION objects. All other standard MPLS and GMPLS objects, including EROs, may be included in messages associated

with this LSP. For example, as viewed at the output of node C in Figure 2, the ingress will be A and the egress could be E. In this case, the ERO sent at A could be [A, B, C, D, E].

A whole P2MP path is encoded via a combination of the standard Explicit Route object (ERO) and a new object that defines each branch. This new object uses the same format as an ERO and is referred to as a Secondary Explicit Route object or SERO, see section 5.2. Multiple SEROs will be used to support a P2MP TE LSP.

At a minimum an SERO will indicate a branch (new ingress) point and a termination (egress) point. Standard ERO semantics can also be used within an SERO to explicitly control the path between branch and termination point.

Nodes at branch points will use SERO information to create a branch LSP. The ERO from the branch LSP will be copied from the relevant SERO. For example, the P2MP path could be represented at A with the ERO [A, B, C, D, E] and the 5 SEROs: SERO #1 [B, J, K, M, N], SERO #2 [K, L], SERO #3 [M, M], SERO #4 [D, F, H, I] and SERO #5 [F, G]. It is worth noting that the first entry of an SERO must always be listed in an LSP's ERO or another SERO.

A Secondary Record Route object or SRRO is also used for recording the path of a branch LSP. In P2MP applications, the SRRO is only present in Resv messages. SRROs are carried from branch points upstream to the ingress. A branching node creates an SRRO by copying the RRO from the Resv message of associated branch LSP into a new SRRO object. Any SRROs present in branch LSP's Resv message are also copied.

The association of all P2P LSPs used to support a P2MP LSP tunnel is enabled via a new object that uniquely identifies the P2MP path. The new object is called the Association object, see section 5.1. All LSPs sharing the same Association object contents are associated with each other. The primary uses for LSP association are management and resource sharing during make-before-break.

4.2. P2MP path calculation

The calculation for a P2MP path requires three major pieces of information. The first is the route from the ingress LSR of a P2MP path to each of the egress LSRs, and the second is the traffic engineering related parameters, including bandwidth etc., on each of the TE links along the route. Note this requirement is exactly the same as calculating a P2P path, except with P2MP there are multiple destination nodes. The third is the branch capability information. Considering a P2MP TE LSP setup over an MPLS network which includes

legacy P2P LSRs, the branch points of the calculated P2MP path should be limited to P2MP capable LSRs.

Routing information and traffic engineering related parameters that are required for calculating a P2MP TE LSP are generally acquired by executing IP routing protocols with TE extensions (for example, OSPF-TE and ISIS-TE). More extensions are necessary to the IP routing protocols to acquire branch capability information. But this is out of the scope of this document.

Using this information, P2MP CSPF calculation function on the ingress LSR or on external path computation server calculates a P2MP path which satisfies several QoS/management constraints. Because several P2MP path calculation algorithms exist and one can implement any kind of algorithm, the specifics of this calculation function is out of the scope of this document.

4.3. P2MP path encoding

The solution divides a calculated P2MP path into a main trunk path and multiple branch paths. A main trunk path is a P2P path which composes the P2MP path and starts from an ingress LSR of the P2MP path and ends with an egress LSR of P2MP path. The main trunk path is encoded as an ERO. A branch path is a P2P path which composes the P2MP path and starts from a branch LSR of the P2MP path and ends with an egress LSR of P2MP path. The branch path is encoded as an SERO, see section 5.2.

```
                             Ingress

                                 A
                                 |
                                 |
                                 B
                                 |
                                 |
                       K----G----C
                       |    |    |
                       |    |    |
                       L    H*   D-------O*
                            |    | \     |
                            |    |  \    |
                            I*   E*  M*  P*
                            |    |   |   |
                            |    |   |   |
                            J*   F   N   Q         X*:P2P only
                                                   Y :P2MP&P2P
                              Egress
```

Figure 3: A P2MP path example

Figure 3. shows a P2MP path example which we want to encode. One feasible approach is utilize depth-first-order dividing and source/branch to egress encoding. If the proposed solution chose a main trunk path as [A, B, C, D, E, F], then depth-first-order algorithm encodes a whole P2MP path as main ERO [A, B, C, D, E, F] and 4 SEROs: SERO #1 [D, M, N], SERO #2 [D, O, P, Q], SERO #3 [C, G, H, I, J] and ERO #4 [G,K,L]. It is worth noting that a main ERO and 4 SEROs are arranged in depth-first-order to indicate the connection of each LSP.

(Following sentences need discussion)

It is possible that make-before-break operations for certain leaf join or prune scenarios will not be possible in networks that support legacy (non-P2MP-capable) LSRs.

The Ingress LSR transmits the ERO and the SEROs in depth-first order. Each LSR that processes the ERO and SEROs maintain the depth-first order. By mandating depth-first order, in a single pass through the ERO & SEROs an LSR can:

- split the ERO & SEROs into groups, one for each next hop LSR
- pop the first subobject of the ERO and each SERO with a first subobject containing a local address
- determine the next hop LSR for each group of ERO/SEROs
- detect whether downstream sub-P2MP LSP is established correctly by

comparing RRO/SRROs with ERO/SEROs.

4.4. Scalability

The proposed solution is scalable because a branch node only handles the same number of path states as the number of downstream neighbors. This means that a branch node does not need to handle path states of all the downstream leaf nodes.

The proposed solution is also scalable because an ingress LSR only needs to send a single path message to operate a P2MP TE LSP. A single path message setup/graft/prune any portion of P2MP path independent of its leaf numbers.

5. P2MP Signalling

5.1. Association Object

The Association object is used to associate LSPs with each other. In the context of this document, the association makes it possible to identify LSPs that support the same P2MP LSP tunnel even when those LSPs belong to different Sessions.

The Association Type, Association Source and Association ID fields of the object together uniquely identify an association. The object uses an object class number of the form 11bbbbbb to ensure compatibility with non-supporting nodes.

5.1.1. Format

The IPv4 Association object has the format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num(TBD)|  C-Type (1)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Association Type        |       Association ID          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  IPv4 Association Source                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The IPv6 Association object has the format:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num(TBD)|  C-Type (2)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Association Type        |       Association ID          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                  IPv6 Association Source                      |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Association Type: 16 bits

Indicates the type of association being identified. Note that this value is considered when determining association. The following are values defined in this document.

Value	Type
0	Reserved
1	Reserved (for Recovery, [RECOVR-SIG])
2	Resource Sharing (R)
3	Multipoint (M)

Association ID: 16 bits

A value that when combined with Association Type and Association Source uniquely identifies an association.

Association Source: 4 or 16 bytes

The IP address of the node that originated the association.

5.1.2. Processing

The Association object is used to associate different LSPs with each other. In the P2MP context, the object is used to associate all the LSPs used to support a P2MP LSP tunnel. It is also used to support resource sharing during make-before-break of P2MP LSP tunnels. The object is carried in Path messages. More than one object may be carried in a single Path message.

Transit nodes MUST transmit, without modification, any received Association objects in the corresponding outgoing Path message(s).

5.1.2.1. Multipoint Association Type Processing

An Association object with an Association Type with the value Multipoint is used to identify a P2MP LSP association.

A node initiating an LSP that is to be used for P2MP MUST insert a Association object with a Multipoint Association Type in the Path message of the P2MP LSP. The Association Source is set to the initiating node's router address. The Association ID MUST be set to the LSP ID of the P2MP LSP.

A node branching a P2MP LSP MUST copy any Association objects of the Multipoint Association Type from the Path message of the associated LSP into the Path message of the new LSP.

Nodes SHOULD use received Association objects which have the Multipoint Association Type to associate LSPs with each other. This association is used to identify a single P2MP path.

5.1.2.2. Resource Sharing Association Type Processing

The Association object with an Association Type with the value Resource Sharing is used to enable resource sharing during make-before-break. Resource sharing during make-before-break is defined in

[RFC3209]. The defined support only works with LSPs that share the same LSP end-point. With the P2MP LSPs this will not always be the case.

A node includes an Association object with a Resource Sharing Association Type in an outgoing Path message when it wishes to indicate resource sharing across an associated set of LSPs. The Association Source is set to the branching node's router address. The Association ID MUST be set to a value that uniquely identifies the association of LSPs. This MAY be set to the upstream LSP's LSP ID. Once included, an Association object with a Resource Sharing Association Type SHOULD NOT be removed from the Path messages associated with an LSP.

When a node is branching an LSP and the associated upstream Path message is received with an Association object with a Resource Sharing type, the branching node inserts an additional Association object with a Resource Sharing type in the Path message of the new LSP. The Association Source is set to the branching node's router address. The Association ID MUST be set to a value that uniquely identifies the association of LSPs. This MAY be set to the upstream LSP's LSP ID.

Any node processing a Path message for a new session which contains an Association object with a Resource Sharing type, examines existing LSPs for matching Association Type, Association Source and Association ID values. If any match is found, then [RFC3209] style resource sharing should be provided between the new and old LSPs. See [RFC3209] for additional details.

5.2. Secondary Explicit Route Object

Secondary Explicit Route objects, or SEROs, are used to indicate the branch points of P2MP LSPs. They may also provide additional information to be carried in a branch LSP's ERO.

5.2.1. Format

The format of a SECONDARY_EXPLICIT_ROUTE object is the same as an EXPLICIT_ROUTE object, Class number 20. This includes the definition of subobjects defined for EXPLICIT_ROUTE object. The class of the SECONDARY_EXPLICIT_ROUTE object is TBA (of form 11bbbbbb).

5.2.2. Path Message Processing

SEROs are carried in Path messages and indicate at which node a branch LSP is to be initiated relative to the LSP carrying the SERO. More than one SERO MAY be present in a Path message.

To indicate the branching nodes of a P2MP LSPs, an SERO is created and added to the Path message of a P2MP LSP. The decision to create and insert an SERO is a local matter and outside the scope of this document.

An SERO SHOULD contain at least two subobjects. The first subobject MUST be a strict hop and MUST indicate the node that is to originate the branch LSP. The address used MUST also be listed in the ERO or another SERO. This ensures that the branch node is along the LSP path. The final subobject in the SERO MUST be the termination point of the branch LSP, and MAY have the L-bit set. Standard ERO subobjects MAY be inserted between the initial subobject and the final subobject. These subobjects MAY be loose or strict.

A node receiving a Path message containing one or more SEROs MUST examine each SERO to see if it indicates a local branch point. This determination is made by examining the first subobject of each SERO and seeing if the address indicated in the subobject is associated with the local node.

If none of the indicated addresses are associated with the local node, then the local node is not a branch node. In this case, all received SEROs MUST be transmitted, without modification, in the corresponding outgoing Path message.

At a branch node, the SERO together with the Path message of LSP being branched provides the information to create the branch LSP. If the processing node is unable to support the requested branch, a PathErr message MUST be sent for the LSP being branched, and normal processing of the LSP continues. The PathErr message SHOULD indicate an error of "TBD" and the Path_State_Removed flag MUST NOT be set. If no error is generated then a branch LSP is created.

The path message for the branch LSP is created by cloning the incoming path message of the LSP being branched. Certain objects are replaced or modified in the new path message. The SENDER_TEMPLATE is updated to use an address on the local node, and the LSP ID is updated to ensure uniqueness. The SESSION object is updated to use the address indicated in the final subobject of the SERO as the tunnel endpoint, the tunnel ID may be updated, and the extended tunnel ID is set to the local node. Any present RROs are cleared of subobjects.

The ERO is replaced with the contents of the SERO that indicated a local branch. The local address and any local subobjects are stripped from the new ERO. The SERO that indicated a local branch is omitted from the new Path message. The list of SEROs on the resulting Path messages should be further reduced as described in section 5.2.2.1.

The resulting Path message is used to create the branch LSP. From this point on, Standard Path message processing is used in processing the resulting Path message.

Note, branch LSPs with loose initial ERO subobjects may be combined. When branch LSPs are combined a new branch node MUST be identified in the outgoing SEROs. Also, any received strict SERO subobjects MUST NOT be modified.

5.2.2.1. Splitting SERO Lists

SEROs SHOULD be omitted, from the new Path message as well as the outgoing Path message for the LSP being branched when the SERO does not relate to the outgoing path message. That is, a Path message SHOULD only contain SEROs that will be used to form sub-P2MP path further down the branch. This is accomplished by only including those SEROs in a Path message when the first subobject of the SERO appears as an object in the ERO of the Path message or in some other SERO that has already been determined suitable for inclusion.

This is an important optimization that reduces the size of Path message within the P2MP path. There is a consequent reduction in transmission of unrelated SEROs in the trigger Path messages to downstream branch and leaf nodes which are not involved in graft/join or prune/leave operation. It is worth noting that it also helps in reducing branch node's operation to detect a corresponding SERO from the received SEROs.

5.2.3. Resv Message Processing

Branch nodes will process Resv messages for both the main upstream LSP and the downstream branch LSPs. A Resv message is propagated upstream of a branch node after a Resv message is received from the first associated downstream LSP, i.e., from either the LSP being branched or from a branch LSP. Subsequent received Resv messages will not typically trigger transmission of upstream Resv messages. Exceptions to this include when RROs are being collected and during certain Admin Status object processing. See below for more information on related Admin Status object processing.

When RROs are collected, each branch node can potentially send a Resv message for each of the downstream receivers. This presents a scalability issue, particularly when considering that the number of messages increases the closer the branch node is to the ingress. 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 SRRO objects, the branch node SHOULD transmit the Resv message only after 100ms has passed since the transmission of the previous Resv message for the same session. This delayed Resv message SHOULD include SRROs for all branches.

5.2.4. 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. Note that when a PathErr message is received by a branch node with the Path_State_Removed flag set (1) and other branch LSPs exist, the downstream portion of that LSP should be torn down, but the whole LSP SHOULD NOT be torn down. The receipt of a PathErr message SHOULD also trigger the generation of a PathErr message upstream on the associated LSP.

This outgoing (upstream) PathErr message SHOULD be sent with the Path_State_Removed flag cleared (0) as only a single branch LSP is impacted. However, if a branch node sends a PathErr message with the Path_State_Removed flag set (1), which is not recommended, the node MUST send a PathTear message downstream on all other branches.

Additionally, an outgoing PathErr message MUST include any SEROs carried in a received PathErr message. If no SERO is present in a received PathErr message, then an SERO that matches the errored LSP MUST be added to the outgoing PathErr message.

5.2.5. Admin Status Change

In general, objects in a branched LSP are created based on the corresponding objects in the LSP being branched. The ADMIN_STATUS object is created the same way, but it also requires some special coordination at branch nodes. Specifically, in addition to normal processing, a branch node that receives an ADMIN_STATUS object also relays the ADMIN_STATUS object in a Path on every branched LSP. All Path messages may be concurrently sent to the downstream neighbors.

Downstream nodes process the change in the 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 on all branches before relaying a corresponding Resv message upstream.

5.2.6. P2MP TE LSP Tear Down

P2MP LSP removal follows standard [RFC3209] and [RFC3473] procedures. This includes with and without setting the administrative status.

See section 5.2.8. for a description of how a branch may be pruned from a P2MP LSP.

5.2.6.1. Tear Down Without Admin Status Change

The ingress originates PathTear message. Each node that receives a PathTear message process the PathTear message as previously defined and also relays a PathTear on every branched LSP. All PathTear messages (received from upstream and locally originated) may be concurrently sent downstream.

5.2.6.2. Tear Down With Admin Status Change

Per [RFC3473], the ingress originates a Path message with the D and R bits in the ADMIN_STATUS object set. The admin status change procedure defined above is then followed. Once the ingress receives all expected Resv messages it follows the tear down procedure described in the previous section.

5.2.6.3. Tear Down From Non-Ingress Nodes

Any node along an LSP branch may initiate removal of the branch. To do this, the node initiating the tear down sends a PathErr with Error Code TBD and the Path_State_Removed flag cleared (0) toward the LSP ingress node. As described above, upstream branch nodes will propagate the error to the LSP ingress which can then signal the removal of the branch, see section 5.2.8.

It is also possible to remove a branch in a non-graceful manner. To do this it simply sends a PathTear downstream and a PathErr with Error Code TBD and the Path_State_Removed flag set(1) toward the LSP ingress. This manner of non-ingress node tear down is NOT RECOMMENDED as it can result in the removal of the entire P2MP TE LSP in some

case.

5.2.7. Grafting

When one or several receivers are to be added to an established P2MP LSP, the ingress originates a Path message with the ERO and SEROs modified as appropriate to represent the modified P2MP path. Each node that receives the message compares the received EROs and SEROs against the previously received objects. When the processing node detects that one or more new branches are present in the message, it handles them using the same method as during P2MP LSP setup. Specifically, it originates one or more branch LSPs as needed to support the P2MP path.

5.2.8. Pruning

When one or several receivers are to be removed from an active P2MP LSP, the ingress originates a Path message with the ERO and SEROs modified as appropriate to represent the modified P2MP path. Each node that receives the message compares the received EROs and SEROs against the previously received objects. When the processing node detects that one or more branches have been removed, it handles them the same method as during P2MP LSP tear down. That is, it sends a PathTear message downstream.

5.2.9. Modification of A P2MP TE LSP

When a P2MP LSP is to be modified, the ingress originates a Path message with the appropriate objects modified to represent the modified P2MP LSP. Each node that receives the message compares the received EROs and SEROs against the previously received objects. When the processing node detects that the ERO or one or more of the SEROs are modified, it originates the Make-Before-Break modification for affected LSPs. Note, that since the branch LSP termination point may be changed, normal Make-Before-Break procedure is not applicable because the SESSION object can not be used as the basis for the resource sharing. The Association object defined above serves as the basis for resource sharing and non-traffic interrupting LSP modification.

5.3. Secondary Record Route Objects

Secondary Record Route objects, or SRROs, are used to record the path used by branch LSPs.

5.3.1. Format

The format of a SECONDARY_RECORD_ROUTE object is the same as an RECORD_ROUTE object, Class number 21. This includes the definition of subobjects defined for RECORD_ROUTE object. The class of the SECONDARY_RECORD_ROUTE object is TBA (of form 11bbbbbb).

5.3.2. Processing

SRROs may be carried in Resv messages and indicate the presence of downstream branches. More than one SRRO MAY be add and present in a Resv message.

Any received SRRO MUST be transmitted by transit nodes, without modification, in the corresponding outgoing Resv message. When Resv messages are merged, the resulting merged Resv SHOULD contain all SRROs received in downstream Resv messages.

SRROs are inserted in Resv messages by the branch node of a P2MP LSP. The SRRO SHOULD be created with the first object being the local node address. The remainder of the SRRO SHOULD be created by copying the contents of the RRO from the received Resv message. This SRRO SHOULD be added to the outgoing Resv message of the branched LSP. Again, multiple SRROs may be present.

If the newly added SRRO causes the message to be too big to fit in a Resv message, SRRO subobjects SHOULD be removed from any present SRROs. When removing subobjects, the first and last subobject in an SRRO MUST NOT be removed. Note that the subobject that followed a removed subobject MUST be updated with the L-bit set (1). If after removing all but the first and last subobjects in all SRROs the resulting message is still too large to fit, then whole SRROs SHOULD be removed until the message does fit.

5.4. Backward Compatibility

The defined approach minimizes backward compatibility issues. When a node that does not support the P2MP extensions receives P2MP Path or Resv message, the newly introduced objects will not be processed and as the objects are the from 11bbbbbb, they will be passed unchanged by non-supporting nodes. Fortunately, this is the exact behavior that is desired from all non-branch nodes. Thus, if a P2MP LSP is required to be delivered by the network, only the ingress and branch nodes must be updated.

6. Updated RSVP Message Formats

This section presents the RSVP message related formats as modified by this document. Where they differ, formats for unidirectional LSPs are presented separately from bidirectional LSPs.

6.1. Path Message Format

The format of a Path message is as follows:

```
<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> ]
                         [ <ASSOCIATION> ... ]
                         [ <SECONDARY_EXPLICIT_ROUTE> ... ]
                         [ <POLICY_DATA> ... ]
                         <sender descriptor>
```

The format of the sender description for unidirectional LSPs is:

```
<sender descriptor> ::=  <SENDER_TEMPLATE> <SENDER_TSPEC>
                         [ <ADSPEC> ]
                         [ <RECORD_ROUTE> ]
                         [ <SUGGESTED_LABEL> ]
                         [ <RECOVERY_LABEL> ]
                         [ <SECONDARY_RECORD_ROUTE> ... ]
```

The format of the sender description for bidirectional LSPs is:

```
<sender descriptor> ::=  <SENDER_TEMPLATE> <SENDER_TSPEC>
                         [ <ADSPEC> ]
                         [ <RECORD_ROUTE> ]
                         [ <SUGGESTED_LABEL> ]
                         [ <RECOVERY_LABEL> ]
                         <UPSTREAM_LABEL>
                         [ <SECONDARY_RECORD_ROUTE> ... ]
```

The format of a PathErr message is as follows:

```
<PathErr Message> ::=    <Common Header> [ <INTEGRITY> ]
                         [ [<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ... ]
                         [ <MESSAGE_ID> ]
                         <SESSION> <ERROR_SPEC>
                         [ <ACCEPTABLE_LABEL_SET> ... ]
                         [ <SECONDARY_EXPLICIT_ROUTE> ... ]
                         [ <POLICY_DATA> ... ]
                         <sender descriptor>
```

6.2. Resv Message Format

The format of a Resv message is as follows:

```
<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> ::= <FLOWSPEC> <FILTER_SPEC>
                           <LABEL> [ <RECORD_ROUTE> ]
                           [ <SECONDARY_RECORD_ROUTE> ... ]
                           | <FF flow descriptor list>
                           <FF flow descriptor>

<FF flow descriptor> ::= [ <FLOWSPEC> ] <FILTER_SPEC> <LABEL>
                           [ <RECORD_ROUTE> ]
                           [ <SECONDARY_RECORD_ROUTE> ... ]

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

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

<SE filter spec> ::=     <FILTER_SPEC> <LABEL> [ <RECORD_ROUTE> ]
                           [ <SECONDARY_RECORD_ROUTE> ... ]
```

7. Application to Traffic Engineering

7.1. Rerouting Traffic Engineered P2MP Tunnels

This protocol supports the make-before-break concept for P2MP TE tunnels.

Both ingress LSR initiated and branch LSR initiated make-before-break operation are supported in this protocol. Detailed operation mechanism will be explained in the future revision.

7.2. Re-establishment of sub-P2MP TE LSP

The graft and prune mechanism can also be used to re-establish a partial P2MP TE LSP when the establishment of the sub-P2MP TE LSP failed. This protocol supports graft and prune operation simultaneously with single Path message by adding and deleting SERO objects.

7.3. P2MP TE tunnel establishment by combining multiple P2MP LSPs

It is possible for very large P2MP paths (that is, those with very many egress nodes) that there is a scaling consideration with the number of SEROs that may be carried in a single Path message.

This concern may be addressed by sending more than one Path message for a single P2MP LSP. Each Path message contains a distinct ERO and a distinct set of SEROs. However, the Association object is used to associate the Path messages and the P2MP LSPs that they form so that resource sharing is achieved on the common legs of the LSPs.

Further, after the first P2MP LSP has been established, a subsequent Path message that adds to the P2MP path may be sent targeted or tunneled to a branch node on the existing P2MP path. This technique avoids upstream nodes needing to maintain duplicate state and avoids issues of resource sharing at legacy transit nodes that do not recognize the Association object.

For example, if we consider to set up a P2MP TE LSP which is explained in Figure 3 by combining two sub-P2MP TE LSPs which share common trunk path [A, B, C, D, E], we must prepare following two sub-P2MP path information. info1: ERO [A, B, C, D, E], SERO #1 [D, M, N] and SERO #2 [D, O, P, Q] info2: ERO [A, B, C, D, E], SERO #1 [C, G, H, I, J], SERO #2 [G, K, L]

Then ingress LSR MUST send multiple path messages which correspond to divided sub-P2MP path and include path information (a common ERO and SEROs) and Association object with Multipoint and Resource Sharing type set.

A P2MP LSP tunnel is supported by multiple sub-P2MP LSPs that may have different Session identifiers and associating these LSPs with Association object. It is worth noting that these LSPs may share resources of common trunk path using the facilities of the Association object.

8. Security Considerations

Security considerations will be addressed in a future revision of this document.

9. Normative References

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

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

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

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

[RFC3473] L. Berger, (Ed.) "Generalized MPLS Signaling - RSVP-TE Extensions", RFC 3473, January 2003.

[P2MP-REQ] S. Yasukawa, (Ed.) "Requirements for Point to Multipoint extension to RSVP-TE", draft-ietf-mpls-p2mp-requirement-01.txt, February 2004, (work in progress).

10. Informational References

[RFC1195] R. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments", RFC 1195, December 1990.

[RFC2328] J. Moy, "OSPF Version 2", RFC 2328, April 1998.

[RFC2702] Awduche, D., Malcolm, J., Agogbua, J., O'Dell and J. McManus, "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

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

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

[RFC3477] Kireeti Kompella, Yakov Rekhter, "Signalling Unnumbered Links in RSVP-TE", RFC3477, January 2003.

[RFC3630] D. Katz, D. Yeung, K. Kompella, "Traffic Engineering Extensions to OSPF Version 2", RFC 3630, September 2003.

[RECOVR-SIG] Lang, J.P., Rekhter, Y., Papadimitriou, D., Editors, "RSVP-TE Extensions in support of End-to-End GMPLS-based Recovery", Work in Progress, draft-lang-ccamp-gmpls-recovery-e2e-signaling-03.txt, February 2004.

[GMPLS-ARCH] Eric Mannie (Ed.), "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", draft-ietf-ccamp-gmpls-architecture-07.txt, May 2003, (work in progress).

[GMPLS-TDM] E. Mannie, D. Papadimitriou, (Eds.) "Generalized Multi-Protocol Label Switching Extensions for SONET and SDH Control", draft-ietf-ccamp-gmpls-sonet-sdh-08.txt, February 2003 (work in progress).

[MPLS-FRR] P. Pan, A. Atlas, (Eds.) "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", draft-ietf-mpls-rsvp-lsp-fastreroute-03.txt, December 2003, (work in progress).

[ISIS-TE] Henk Smit, Tony Li, "IS-IS extensions for Traffic Engineering", draft-ietf-isis-traffic-05.txt, August 2003, (work in progress).

11. Contributors

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

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

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

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

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

12. Editor's Address

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

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

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

13. Intellectual Property Consideration

14. Full Copyright Statement

Appendix. Differences between RSVP Multicasting and P2MP TE Tunnels

The applications as supported by the protocol extensions described in this document are different from RSVP multicast applications as described in the [RFC2205]. In general, their differences are similar to that between RSVP-TE tunnels as described in the [RFC3209] and RSVP point-to-point applications as described in the [RFC2205].

One of the key differences is that the P2MP path is explicitly specified at the sender where a main ERO and SEROs are used that is similar semantically to the ERO as specified in the [RFC3209], whereas in the RSVP multicasting case, the multicasting tree is constructed hop-by-hop at every tree node based on the routing information collected from multicast routing protocols.

One of the other key differences is that the RSVP multicasting was defined only for applications in IP networks, while the RSVP-TE P2MP connections as defined in this document are for MPLS applications in IP networks as well as in non-IP networks where the GMPLS technology

applies per [GMPLS-ARCH]. In particular, the separation of the control plane and data plane, which is one of the important building blocks in the GMPLS architecture, is inherited in this document to support a natural extension of the GMPLS-RSVP ([RFC3473]), i.e., the P2MP TE tunnels.

Because the RSVP P2MP TE tunnels as specified in this document can be seen as an extension to the RSVP P2P TE tunnels defined in MPLS ([RFC3209]) and GMPLS ([RFC3473]) networks, all the traffic engineering related objects and features as defined in the MPLS/GMPLS might also be used to support RSVP P2MP tunnels, including the following:

- Label object and its technology-dependent encoding
- Unnumbered links
- Admin status object
- Protection object
- Etc.

Note that all of these objects and features are not applicable for RSVP sessions using unicast or multicast destination addresses as defined in the [RFC2205], but are specifically defined for P2P MPLS/GMPLS TE tunnels, and now for P2MP tunnels as described in this document.