Network Working Group
Request for Comments: 4873
Updates: 3473, 4872
Category: Standards Track

L. Berger
LabN Consulting
I. Bryskin
ADVA Optical
D. Papadimitriou
Alcatel
A. Farrel
Old Dog Consulting

May 2007

GMPLS Segment Recovery

Status of This Memo

Copyright Notice

Abstract

This document describes protocol specific procedures for GMPLS (Generalized Multi-Protocol Label Switching) RSVP-TE (Resource ReserVation Protocol - Traffic Engineering) signaling extensions to support label switched path (LSP) segment protection and restoration. These extensions are intended to complement and be consistent with the RSVP-TE Extensions for End-to-End GMPLS Recovery (RFC 4872). Implications and interactions with fast reroute are also addressed. This document also updates the handling of NOTIFY_REQUEST objects.

Table of Contents

1. Introduction

[RFC4427] covers multiple types of protection, including end-to-end and segment-based approaches. "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery" [RFC4872] defines a set of extensions to support multiple types of recovery. The supported types include 1+1 unidirectional/ 1+1 bidirectional protection, LSP protection with extra-traffic (including 1:N protection with extra-traffic), pre-planned LSP re-routing without extra-traffic (including shared mesh), and full LSP re-routing. In all cases, the recovery is provided on an end-to-end basis, i.e., the ingress and egress nodes of both the protected and the protecting LSP are the same.

[RFC4090] provides a form of segment recovery for packet MPLS-TE networks. Two methods of fast reroute are defined in [RFC4090]. The one-to-one backup method creates detour LSPs for each protected LSP at each potential point of local repair. The facility backup method creates a bypass tunnel to protect a potential failure point that is shared by multiple LSPs and uses label stacking. Neither approach supports the full set of recovery types supported by [RFC4872]. Additionally, the facility backup method is not applicable to most non-PSC (packet switch capable) switching technologies.

The extensions defined in this document allow for support of the full set of recovery types supported by [RFC4872], but on a segment, or portion of the LSP, basis. The extensions allow (a) the signaling of desired LSP segment protection type, (b) upstream nodes to optionally identify where segment protection starts and stops, (c) the optional identification of hops used on protection segments, and (d) the reporting of paths used to protect an LSP. The extensions also widen the topological scope over which protection can be supported. They allow recovery segments that protect against an arbitrary number of nodes and links. They enable overlapping protection and nested protection. These extensions are intended to be compatible with fast reroute, and in some cases used with fast reroute.

1.1. Conventions Used in This Document

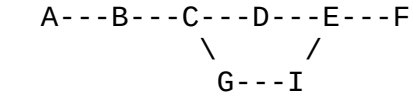
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In addition, the reader is assumed to be familiar with the terminology used in [RFC3209], [RFC3471], and [RFC3473], as well as [RFC4427], [RFC4426], [RFC4872], and [RFC4090].

2. Segment Recovery

Segment recovery is used to provide protection and restoration over a portion of an end-to-end LSP. Such segment protection and restoration is useful to protect against a span failure, a node failure, or failure over a particular portion of a network used by an LSP.

Consider the following topology:



In this topology, end-to-end protection and recovery is not possible for an LSP going between node A and node F, but it is possible to protect/recover a portion of the LSP. Specifically, if the LSP uses a working path of [A,B,C,D,E,F], then a protection or restoration LSP can be established along the path [C,G,I,E]. This LSP protects against failures on spans {C,D} and {D,E}, as well as a failure of node D. This form of protection/restoration is referred to as Segment Protection and Segment Restoration, or as Segment Recovery, collectively. The LSP providing the protection or restoration is referred to as a segment protection LSP or a segment restoration LSP. The term "segment recovery LSP" is used to cover either a segment protection LSP or a segment restoration LSP. The term "branch node" is used to refer to a node that initiates a recovery LSP, e.g., node C in the figure shown above. This is equivalent to the point of local repair (PLR) used in [RFC4090]. As with [RFC4090], the term "merge node" is used to refer to a node that terminates a recovery LSP, e.g., node E in the figure shown above.

Segment protection or restoration is signaled using a working LSP and one or more segment recovery LSPs. Each segment recovery LSP is signaled as an independent LSP. Specifically, the Sender_Template object uses the IP address of the node originating the recovery path, e.g., node C in the topology shown above, and the Session object contains the IP address of the node terminating the recovery path, e.g., node E shown above. There is no specific requirement on LSP ID value, Tunnel ID, and Extended Tunnel ID. Values for these fields are selected normally, including consideration for the make-before-break concept (as described in [RFC3209]). Intermediate nodes follow standard signaling procedures when processing segment recovery LSPs. A segment recovery LSP may be protected itself using segment or end-to-end protection/restoration. Note, in PSC environments, it may be desirable to construct the Sender_Template and Session objects per [RFC4090].

When [RFC4090] isn't being used, the association between segment recovery LSPs with other LSPs is indicated using the ASSOCIATION object defined in [RFC4872]. The ASSOCIATION object is used to associate recovery LSPs with the LSP they are protecting. Working and protecting LSPs, as well as primary and secondary LSPs, are identified using LSP Status as described in [RFC4872]. The O-bit in the segment flags portion of the PROTECTION object is used to identify when a recovery LSP is carrying the normal (active) traffic.

An upstream node can permit downstream nodes to dynamically identify branch and merge points by setting the desired LSP segment protection bits in the PROTECTION object. These bits are defined below.

Optionally, an upstream node, usually the ingress node, can identify the endpoints of a segment recovery LSP. This is accomplished using a new object. This object uses the same format as an Explicit Route Object (ERO) and is referred to as a Secondary Explicit Route object (SERO); see Section 4.1. SEROs also support a new subobject to indicate the type of protection or restoration to be provided. At a minimum, an SERO will indicate a recovery LSP's initiator, protection/restoration type and terminator. Standard ERO semantics (see [RFC3209]) can optionally be used within and SERO to explicitly control the recovery LSP. A Secondary Record Route object (SRRO) is defined for recording the path of a segment recovery LSP; see Section 5.

SEROs are carried between the node creating the SERO, typically the ingress, and the node initiating a recovery LSP. The node initiating a recovery LSP uses the SERO to create the ERO for the recovery LSP. At this (branch) node, all local objects are removed, and the new protection subobject is used to create the PROTECTION object for the recovery LSP. It is also possible to control the handling of a failure to establish a recovery LSP.

SRROs are carried in Path messages between the node terminating a recovery LSP, the merge node, and the egress. SRROs are used in Resv messages between a branch node and the ingress. The merge node of a recovery LSP creates an SRRO by copying the RRO from the Path message of the associated recovery LSP into a new SRRO object. Any SRROs present in the recovery LSP's Path message are also copied. The branch node of a recovery LSP creates an SRRO by copying the RRO from the Resv message of associated recovery LSP into a new SRRO object. Any SRROs present in the recovery LSP's Resv message are also copied.

Notify request processing is also impacted by LSP segment recovery. Per [RFC3473], only one NOTIFY_REQUEST object is meaningful and should be propagated. Additional NOTIFY_REQUEST objects are used to identify recovery LSP branch nodes.

2.1. Segment Protection

Three approaches for end-to-end protection are defined in [RFC4872]: 1+1 Unidirectional Protection (Section 5), 1+1 Bidirectional Protection (Section 6), and 1:1 Protection With Extra-Traffic (Section 7). The segment protection forms of these protection approaches all operate much like their end-to-end counterparts. Each behaves just like its end-to-end counterpart, with the exception that the protection LSP protects only a portion of the working LSP. The type of protection to be used on a segment protection LSP is indicated, to the protection LSP's ingress, using the protection SERO subobject defined in Section 4.1.

The switch-over processing for segment 1+1 Bidirectional protection and 1:1 Protection With Extra-Traffic follows the same procedures as end-to-end protection forms; see Sections 6.2 and 7.2 of [RFC4872] for details.

2.2. Segment Re-routing and Restoration

Three re-routing and restoration approaches are defined in [RFC4872]: Re-routing without Extra-Traffic (Section 8), Shared-Mesh Restoration (Section 9), (Full) LSP Re-routing (Section 11). As with protection, these approaches are supported on a segment basis. The segment forms of re-routing and restoration operate exactly like their end-to-end counterparts, with the exception that the restoration LSP recovers only a portion of the working LSP. The type of re-routing or restoration to be used on a segment restoration LSP is indicated, to the restoration LSP's ingress, using the new protection SERO subobject.

3. ASSOCIATION Object

The ASSOCIATION object is used for the association of segment protection LSPs when [RFC4090] isn't being used. The ASSOCIATION object is defined in [RFC4872]. In this document, we define a new Association Type field value to support make-before-break; the formats and procedures defined in [RFC4872] are not otherwise modified.

3.1. Format

Association Type: 16 bits

Value	Type
2	Resource Sharing (R)

See [RFC4872] for the definition of other fields and values.

3.2. Procedures

The ASSOCIATION object is used to associate different LSPs with each other. In the protection and restoration context, the object is used to associate a recovery LSP with the LSP it is protecting. The ASSOCIATION object is also used to support resource sharing during make-before-break. This object MUST NOT be used when association is made according to the methods defined in [RFC4090].

3.2.1. Recovery Type Processing

Recovery type processing procedures are the same as those defined in [RFC4872], but processing and identification occur with respect to segment recovery LSPs. Note that this means that multiple ASSOCIATION objects of type recovery may be present on an LSP.

3.2.2. 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 egress. With the introduction of segment recovery LSPs, it is now possible for an LSP endpoint to change during make-before-break.

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 originating 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 working 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.

Any node processing a Path message for which the node does not have a matching state, and 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.

4. Explicit Control of LSP Segment Recovery

Secondary Explicit Route objects, or SEROs, are defined in this document. They may be used to indicate the branch and merge nodes of recovery LSPs. They may also provide additional information that is to be carried in a recovery LSP's ERO. When upstream control of branch and merge nodes is not desired, SEROs are not used.

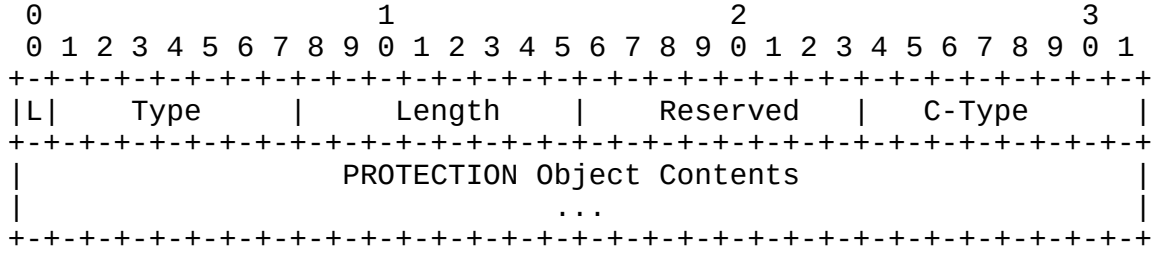
4.1. Secondary Explicit Route Object Format

The format of a SECONDARY_EXPLICIT_ROUTE object is the same as an EXPLICIT_ROUTE object. This includes the definition of subobjects defined for EXPLICIT_ROUTE object. The class of the SECONDARY_EXPLICIT_ROUTE object is 200 (of the form 11bbbbbb).

4.1.1. Protection Subobject

A new subobject, called the protection subobject, is defined for use in the SECONDARY_EXPLICIT_ROUTE object. As mentioned above, the new protection subobject is used to create the PROTECTION object for the recovery LSP. Specific procedures related to the protection subobject are provided in Section 4.2. The protection subobject is not valid for use with the Explicit and Record Route objects and MUST NOT be included in those objects.

The format of the protection subobject is defined as follows:



L-bit

This is defined in [RFC3209] and MUST be set to zero for protection subobjects.

Type

37 Protection

Length

As defined in [RFC3209], Section 4.3.3.

Reserved

This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

C-Type

The C-Type of the included PROTECTION object.

PROTECTION Object Contents

The contents of the PROTECTION object, with the format matching the indicated C-Type, excluding the object header.

4.2. Explicit Control Procedures

SEROs are carried in Path messages and indicate at which node a recovery 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 branch and merge nodes of a recovery LSP, an SERO is created and added to the Path message of the LSP being recovered. 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 three subobjects. The first subobject MUST indicate the node that is to originate the recovery LSP, i.e. the segment branch node. The address used SHOULD also be listed in the ERO or another SERO. This ensures that the branch node is along the LSP path. The second subobject SHOULD be a protection subobject and should indicate the protection or restoration to be provided by the recovery LSP. When the protection subobject is present, the LSP Segment Recovery Flags in the protection subobject MUST be ignored. The final subobject in the SERO MUST be the merge node of the recovery LSP, and MAY have the L-bit set. Standard ERO subobjects MAY be inserted between the protection subobject and the final subobject. These subobjects MAY be loose or strict.

A node receiving a Path message containing one or more SEROs SHOULD examine each SERO to see if it indicates a local branch point. This

determination is made by examining the first object of each SERO and seeing if the address indicated in the subobject can be associated with the local node. If any of indicated addresses are associated with the local node, then the local node is a branch node. If the local node is not a branch node, 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 recovered, provides the information to create the recovery LSP. The Path message for the recovery LSP is created at the branch node by cloning the objects carried in the incoming Path message of the LSP being protected. Certain objects are replaced or modified in the recovery LSP's outgoing Path message. The Sender_template object MUST be updated to use an address (in its Tunnel Sender Address field) on the local node, and the LSP ID MUST be updated to ensure uniqueness. The Session object MUST be updated to use the address indicated in the final subobject of the SERO as the tunnel endpoint address, the tunnel ID MAY be updated, and the extended tunnel ID MUST be set to the local node address. The PROTECTION object is replaced with the contents of the matching SERO protection subobject, when present. In all cases, the R-bit of a new PROTECTION object is reset (0). Any RROs and EROs present in the incoming Path message MUST NOT be included in the recovery LSP. A new ERO MUST be included, with the contents of the SERO that indicated a local branch. As with all EROs, no local information (local address and any protection subobjects) is carried in the ERO carried in the recovery LSP's outgoing Path message. The SERO that indicated a local branch MUST be omitted from the recovery LSP's outgoing Path message. Note, by default, all other received SEROs are passed in the recovery LSP's outgoing Path message. SEROs MAY be omitted, from the recovery LSP's outgoing Path message as well as the outgoing Path message for the LSP being protected, when the SERO does not relate to the outgoing path message.

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

4.2.1. Branch Failure Handling

During setup, it is possible that a processing node will be unable to support a requested branch. Additionally, during setup and normal operation, PathErr messages may be received at a branch node. The processing of these events depend on a number of factors.

When a failure or received PathErr message is associated with the LSP being protected, the event is first processed per standard processing

rules. This includes generation of a standard PathErr message. When LSP state is removed due to a local failure or a PathErr message with the Path_State_Removed flag set (1), the node MUST send a PathTear message downstream on all other branches.

When a failure or received PathErr message is associated with a recovery LSP, processing is based on the R-bit in addition to the Path_State_Removed flag. In all cases, a received PathErr message is first processed per standard processing rules and the failure or received PathErr message SHOULD trigger the generation of a PathErr message upstream for the LSP being protected. The outgoing PathErr message SHOULD indicate an error of "Routing Problem/LSP Segment Protection Failed". The outgoing PathErr message MUST include any SEROs carried in a received PathErr message. If no SERO is present in a received PathErr message or when the failure is local, then an SERO that matches the errored LSP or failed branch MUST be added to the outgoing PathErr message.

When a PathErr message with the Path_State_Removed flag cleared (0) is received, the outgoing (upstream) PathErr message SHOULD be sent with the Path_State_Removed flag cleared (0).

When a PathErr message for a recovery LSP with the Path_State_Removed flag set (1) is received, the processing node MUST examine the R-bit (as defined below) of the LSP being protected. The R-bit is carried in the PROTECTION object that triggered the initiation of the recovery LSP. When the R-bit is not set (0), the outgoing (upstream) PathErr message SHOULD be sent with the Path_State_Removed flag cleared (0). When the R-bit is set (1), the outgoing (upstream) PathErr message MUST be sent with the Path_State_Removed flag set (1).

In all cases where an outgoing (upstream) PathErr message is sent with the Path_State_Removed flag set (1), all path state for the LSP being protected MUST be removed, and the node MUST send a PathTear message downstream on all active branches.

4.2.2. Resv Message Processing

Branch nodes will process Resv messages for both recovery LSPs and LSPs being protected. Resv messages are propagated upstream of branch nodes only after a Resv message is received for the protected LSP. Resv messages on recovery LSPs will typically not trigger transmission of upstream Resv messages (for the LSP being protected). Exceptions to this include when RROs/SRROs are being collected and during certain ADMIN_STATUS object processing. See below for more information on related processing.

4.2.3. Admin Status Change

In general, objects in a recovery LSP are created based on the corresponding objects in the LSP being protected. 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 in a Path message also MUST relay the ADMIN_STATUS object in a Path on every recovery LSP. All Path messages MAY be concurrently sent downstream.

Downstream nodes process the change in the ADMIN_STATUS object per [RFC3473], including generation of Resv messages. When the most recently received upstream ADMIN_STATUS object has 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.

4.2.4. Recovery LSP Teardown

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

4.2.4.1. Teardown Without Admin Status Change

The node initiating the teardown originates a PathTear message. Each node that receives a PathTear message processes the PathTear message per standard processing (see [RFC3209] and [RFC2205]), and MUST also relay a PathTear on every recovery LSP. All PathTear messages (received from upstream and locally originated) may be concurrently sent downstream.

4.2.4.2. Teardown With Admin Status Change

Per [RFC3473], the ingress node originates a Path message with the D and R bits set in the ADMIN_STATUS object. The admin status change procedure defined in Section 4.2.3 MUST then be followed. Once the ingress receives all expected Resv messages, it MUST follow the teardown procedure described in Section 4.2.4.1.

4.3. Teardown From Non-Ingress Nodes

As with any LSP, any node along a recovery LSP may initiate removal of the recovery LSP. To do this, the node initiating the teardown sends a PathErr message with the appropriate Error Code and the Path_State_Removed flag cleared (0) toward the LSP ingress. As described above, the recovery LSP ingress will propagate the error to

the LSP ingress, which can then signal the removal of the recovery LSP.

It is also possible for the node initiating the teardown to remove a Recovery LSP in a non-graceful manner. In this case, the initiator sends a PathTear message downstream and a PathErr message with a "Confirmation" indication (error code and value set to zero), and the Path_State_Removed flag set (1) toward the LSP ingress node. This manner of non-ingress node teardown is NOT RECOMMENDED because in some cases it can result in the removal of the LSP being protected.

4.3.1. Modified NOTIFY_REQUEST Object Processing

A parallel set of rules are applied at branch and merge nodes to enable the branch or merge node to add a NOTIFY_REQUEST object to the Path and Resv messages of protected and recovery LSPs. Branch nodes add NOTIFY_REQUEST objects to Path messages, and merge nodes add NOTIFY_REQUEST objects to Resv messages.

When a node is branching or merging a recovery LSP, the node SHOULD include a single NOTIFY_REQUEST object in the Path message of the recovery LSP, in the case of a branch node, or the Resv message of the recovery LSP, in the case of a merge node. The notify node address MUST be set to the router address of the processing node.

Branch and merge nodes SHOULD also add a NOTIFY_REQUEST object to the LSP being protected. For branch nodes, the notify node address is set to the address used in the sender template object of the associated recovery LSP. For merge nodes, the notify node address is set to the address used in the session object of the associated recovery LSP. A locally added NOTIFY_REQUEST object MUST be listed first in an outgoing message; any received NOTIFY_REQUEST objects MUST then be listed in the message in the order that they were received. Note that this can result in a stack (or ordered list) of objects.

Normal notification procedures are then followed for the LSP being protected. That is, the notifying node MUST issue a Notify message to the recipient indicated by the notify address of the first listed NOTIFY_REQUEST object. Under local policy control, a node issuing a Notify message MAY also send a Notify message to the Notify Node Address indicated in the last, or any other, NOTIFY_REQUEST object carried in the Path or Resv message.

Recovery LSP merge and branch nodes remove the object added by the recovery branch or merge node from outgoing Path and Resv messages for the LSP being protected. This is done by removing the NOTIFY_REQUEST object that, in the case of a merge node, matches the

source address of the recovery LSP and, in the case of a branch node, matches the tunnel endpoint address of the recovery LSP. The matching NOTIFY_REQUEST object will normally be the first of the listed NOTIFY_REQUEST objects. Note, to cover certain backwards compatibility scenarios, the NOTIFY_REQUEST object SHOULD NOT be removed if it is the sole NOTIFY_REQUEST object.

Note this requires the following change to [RFC3473], Section 4.2.1:

o old text:
   If a message contains multiple NOTIFY_REQUEST objects, only the first object is meaningful. Subsequent NOTIFY_REQUEST objects MAY be ignored and SHOULD NOT be propagated.

o new text:
   If a message contains multiple NOTIFY_REQUEST objects, only the first object used is in notification. Subsequent NOTIFY_REQUEST objects MUST be propagated in the order received.

4.3.2. Modified Notify and Error Message Processing

Branch and merge nodes MUST support the following modification to Notify message processing. When a branch or merge node receives notification of an LSP failure and it is unable to recover from that failure, it MUST notify the node indicated in the first NOTIFY_REQUEST object received in the Path message (for branch nodes) or Resv message (for merge nodes) associated with the LSP.

5. Secondary Record Route Objects

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

5.1. Format

The format of a SECONDARY_RECORD_ROUTE object is the same as a 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 201 (of the form 11bbbbbb).

The protection subobject defined above can also be used in SECONDARY_RECORD_ROUTE objects.

5.2. Path Processing

SRROs may be carried in Path messages and indicate the presence of upstream recovery LSPs. More than one SRRO MAY be added and present in a Path message.

Any received SRRO MUST be transmitted by transit nodes, without modification, in the corresponding outgoing Path message.

SRROs are inserted in Path messages by recovery LSP merge nodes. The SRRO is created by copying the contents of an RRO received by the recovery LSP into a new SRRO object. This SRRO is added to the outgoing Path message of the recovered LSP. Note that multiple SRROs may be present. The collection of SRROs is controlled via the segment-recording-desired flag in the SESSION_ATTRIBUTE object. This flag MAY be set even when SEROs are not used.

5.3. Resv Processing

SRROs may be carried in Resv messages and indicate the presence of downstream recovery LSPs. More than one SRRO MAY be added 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 branch nodes of recovery LSPs. The SRRO SHOULD be created with the first two objects being the local node address, followed by a protection subobject with the contents of the recovery LSP's PROTECTION object. The remainder of the SRRO SHOULD be created by copying the contents of the RRO received by the recovery LSP. This SRRO SHOULD be added to the outgoing Resv message of the recovered 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 two subobjects and the 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.

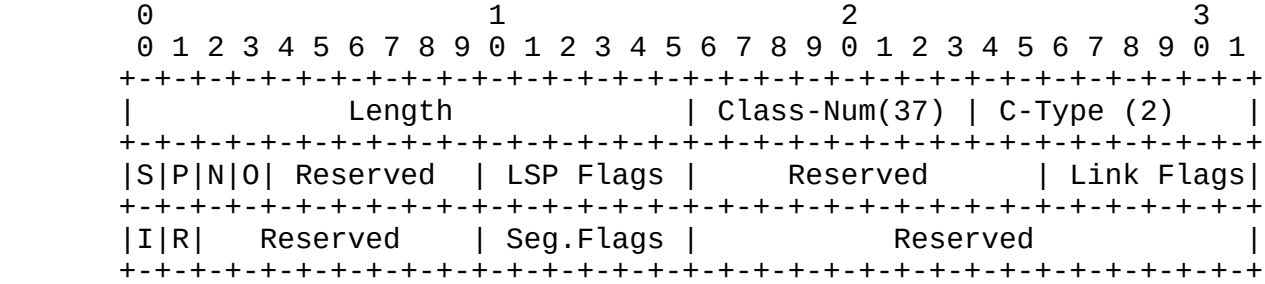
6. Dynamic Control of LSP Segment Recovery

Dynamic identification of branch and merge nodes is supported via the LSP Segment Recovery Flags carried in the PROTECTION object. The LSP Segment Recovery Flags are carried within one of the Reserved fields defined in the PROTECTION object defined in [RFC4872]. LSP Segment Recovery Flags are used to indicate when LSP segment recovery is desired. When these bits are set, branch and merge nodes are dynamically identified.

Note, the procedures defined in this section parallel the explicit control procedures defined above in Section 4.2. The primary difference is in the creation of a recovery LSP's ERO.

6.1. Modified PROTECTION Object Format

LSP Segment Recovery Flags are carried in the PROTECTION object of the same C-Type defined in [RFC4872]. The format of the flags are:



In-Place (I): 1 bit

When set (1) indicates that the desired segment recovery type indicated in the LSP Segment Recovery Flag is already in place for the associated LSP.

Required (R): 1 bit

When set (1) indicates that failure to establish the indicated protection should result in a failure of the LSP being protected.

Segment Recovery Flags (Seg.Flags): 6 bits

This field is used to indicate when an upstream node desires LSP Segment recovery to be dynamically initiated where possible. The values used in this field are identical to the values defined for LSP Flags; see [RFC4872].

See [RFC4872] for the definition of other fields.

6.2. Dynamic Control Procedures

LSP Segment Recovery Flags are set to indicate that LSP segment recovery is desired for the LSP being signaled. The type of recovery desired is indicated by the flags. The decision to set the LSP Segment Recovery Flags is a local matter and outside the scope of this document. A value of zero (0) means that no dynamic identification of segment recovery branch nodes are needed for the associated LSP. When the In-Place bit is set, it means that the desired type of recovery is already in place for that particular LSP.

A transit node receiving a Path message containing a PROTECTION object with a non-zero LSP Segment Recovery Flags value and the In-Place bit clear (0) SHOULD consider if it can support the indicated recovery type and if it can identify an appropriate merge node for a recovery LSP. Dynamic identification MUST NOT be done when the processing node is identified as a branch node in an SERO. If a node is unable to provide the indicated recovery type or identify a merge node, the Path message MUST be processed normally, and the LSP Segment Recovery Flags MUST NOT be modified.

When a node dynamically identifies itself as a branch node and identifies the merge node for the type of recovery indicated in the LSP Segment Recovery Flags, it attempts to setup a recovery LSP. The dynamically identified information, together with the Path message of LSP being recovered, is used to create the recovery LSP.

The Path message for the recovery LSP is created at the branch node by cloning the objects carried in the incoming Path message of the LSP being protected. Certain objects are replaced or modified in the recovery LSP's outgoing Path message. The Sender_template object MUST be updated to use an address (in its Tunnel Sender Address field) on the local node, and the LSP ID MUST be updated to ensure uniqueness. The Session object MUST be updated to use the address of the dynamically identified merge node as the tunnel endpoint address, the tunnel ID MAY be updated, and the extended tunnel ID MUST be set to the local node address. The PROTECTION object is updated with the In-Place bit set (1). Any RROs and EROs present in the incoming Path message MUST NOT be included in the recovery LSP. A new ERO MAY be created based on any path information dynamically computed by the local node.

The resulting Path message is used to create the recovery LSP. While the recovery LSP exists, the PROTECTION object in the original Path message (unless overridden by local policy) MUST also be updated with the In-Place bit set (1). From this point on, Standard Path message processing is used in processing the resulting and original Path messages.

The merge node of a dynamically controlled recovery LSP SHOULD reset (0) the In-Place bit in the PROTECTION object of the outgoing Path message associated with the terminated recovery LSP.

Unlike with explicit control, if the creation of a dynamically identified recovery LSP fails for any reason, the recovery LSP is removed, and no error message or indication is sent upstream. With this exception, all the other procedures for explicitly controlled recovery LSPs apply to dynamically controlled recovery LSPs. These other procedures are defined above in Sections 4.2.1 through 4.2.4.

7. 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.

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>
```

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> ... ]
```

8. Security Considerations

This document introduces new message objects for use in GMPLS signaling [RFC3473]. It does not introduce any new signaling messages, nor change the relationship between LSRs that are adjacent in the control plane.

The procedures defined in this document result in an increase in the amount of topology information carried in signaling messages since the presence of SEROs and SRROs necessarily means that there is more information about LSP paths carried than in simple EROs and RROs. Thus, in the event of the interception of a signaling message, slightly more could be deduced about the state of the network than was previously the case, but this is judged to be a very minor security risk as this information is already available via routing.

Otherwise, this document introduces no additional security considerations. See [RFC3473] for relevant security considerations.

9. IANA Considerations

IANA has assigned the following values for the namespaces defined in this document and reviewed in this section.

9.1. New Association Type Assignment

IANA has made the following assignment to the "GMPLS Signaling Parameters" Registry (see [RFC4872]) located at http://www.iana.org/assignments/gmpls-sig-parameters.

Value	Type
2	Resource Sharing (R) [RFC4873]

9.2. Definition of PROTECTION Object Reserved Bits

This document defines bits carried in the Reserved field of the PROTECTION object defined in [RFC4872]. As no IANA registry for these bits is requested in [RFC4872], no IANA action is required related to this definition.

9.3. Secondary Explicit Route Object

IANA has made the following assignments in the "Class Names, Class Numbers, and Class Types" section of the "RSVP PARAMETERS" registry located at http://www.iana.org/assignments/rsvp-parameters.

A new class named SECONDARY_EXPLICIT_ROUTE has been created in the 11bbbbbb range (200) with the following definition:

Class Types or C-types:

Same values as EXPLICIT_ROUTE object (C-Num 20)

For Class 1, C-Type 1, the following additional Subobject type is defined:

37 PROTECTION [RFC4873]

9.4. Secondary Record Route Object

IANA has made the following assignments in the "Class Names, Class Numbers, and Class Types" section of the "RSVP PARAMETERS" registry located at http://www.iana.org/assignments/rsvp-parameters.

A new class named SECONDARY_RECORD_ROUTE has been created in the 11bbbbbb range (201) with the following definition:

Class Types or C-types:

Same values as RECORD_ROUTE object (C-Num 21)

For Class 1, C-Type 1, the following additional Subobject type is defined:

37 PROTECTION [RFC4873]

9.5. New Error Code

IANA has made the following assignments in the "Routing Problem" subsection of "Error Codes and Values" section of the "RSVP PARAMETERS" registry located at http://www.iana.org/assignments/rsvp-parameters.

21 = LSP Segment Protection Failed [RFC4873]

9.6. Use of PROTECTION Object C-type

This document modifies the PROTECTION object, class number 37, C-Type 2 (defined in Section 14.1. of [RFC4872]).

10. References

10.1. 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.

[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.

[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.

[RFC4872] Lang, J.P., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, May 2007.

10.2. Informative References

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

[RFC4426] Lang, J., Ed., Rajagopalan, B., Ed., and D. Papadimitriou, Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Recovery Functional Specification," RFC 4426, March 2006.

[RFC4427] Mannie, E., Ed., and D. Papadimitriou, Ed., "Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4427, March 2006.

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Phone: +1 301-468-9228
EMail: lberger@labn.net

Igor Bryskin
ADVA Optical
7926 Jones Branch Drive
Suite 615
McLean VA, 22102

EMail: IBryskin@advaoptical.com

Dimitri Papadimitriou
Alcatel
Francis Wellesplein 1
B-2018 Antwerpen, Belgium

Phone: +32 3 240-8491
EMail: dimitri.papadimitriou@alcatel-lucent.be

Adrian Farrel
Old Dog Consulting

Phone: +44 (0) 1978 860944
EMail: adrian@olddog.co.uk

Full Copyright Statement

Intellectual Property

Acknowledgement

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