Network Working Group
Internet Draft
Expiration Date: October 2002

Lou Berger - Editor (Movaz Networks)
Peter Ashwood-Smith (Nortel Networks)
Ayan Banerjee (Calient Networks)
Greg Bernstein (Ciena Corporation)
John Drake (Calient Networks)
Yanhe Fan (Axiowave Networks)
Kireeti Kompella (Juniper Networks)
Jonathan P. Lang (Calient Networks)
Fong Liaw (Solas Research)
Eric Mannie (KPNQwest)
Ping Pan (Juniper Networks, Inc.)
Bala Rajagopalan (Tellium)
Yakov Rekhter (Juniper Networks)
Debanjan Saha (Tellium)
Vishal Sharma (Metanoia)
George Swallow (Cisco Systems)
Z. Bo Tang (Tellium)

April 2002

Generalized MPLS Signaling - RSVP-TE Extensions

draft-ietf-mpls-generalized-rsvp-te-07.txt

Status of this Memo

Abstract

This document describes extensions to RSVP-TE signaling required to support Generalized MPLS. Generalized MPLS extends MPLS to encompass time-division (e.g. SONET/SDH ADMs), wavelength (optical lambdas) and spatial switching (e.g. incoming port or fiber to outgoing port or fiber). This document presents an RSVP-TE specific description of the extensions. A generic functional description and a CR-LDP specific description can be found in separate documents.

Contents

[Editor's note: changes to be removed prior to publication as an RFC.]
Changes from previous version:

- Reorder author's list to indicate primary contact
- Fixed indication of infinite restart time
- Added IANA [Suggestions /] Assignments Section
- Split references section
- Minor editorial changes and clarifications

1. Introduction

Generalized MPLS extends MPLS from supporting packet (PSC) interfaces and switching to include support of three new classes of interfaces and switching: Time-Division Multiplex (TDM), Lambda Switch (LSC) and Fiber-Switch (FSC). A functional description of the extensions to MPLS signaling needed to support the new classes of interfaces and switching is provided in [GMPLS-SIG]. This document presents RSVP-TE specific formats and mechanisms needed to support all four classes of interfaces. CR-LDP extensions can be found in [GMPLS-LDP].

[GMPLS-SIG] should be viewed as a companion document to this document. The format of this document parallels [GMPLS-SIG]. In addition to the other features of Generalized MPLS, this document also defines RSVP-TE specific features to support rapid failure notification, see Sections 4.2 and 4.3.

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].

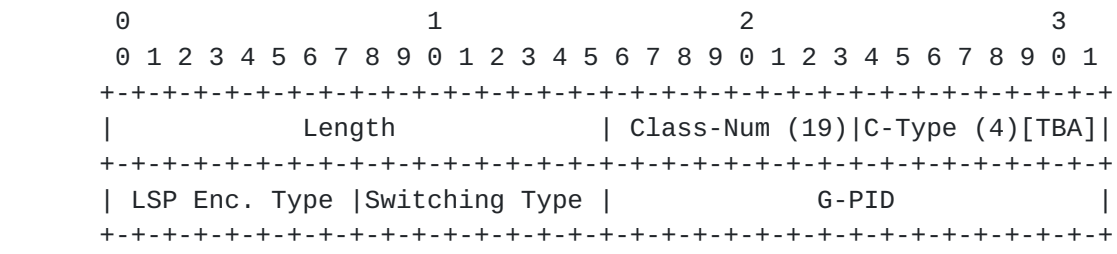
2. Label Related Formats

This section defines formats for a generalized label request, a generalized label, support for waveband switching, suggested label and label sets.

2.1. Generalized Label Request Object

A Path message SHOULD contain as specific an LSP Encoding Type as possible to allow the maximum flexibility in switching by transit LSRs. A Generalized Label Request object is set by the ingress node, transparently passed by transit nodes, and used by the egress node. The Switching Type field may also be updated hop-by-hop.

The format of a Generalized Label Request object is:



See [GMPLS-SIG] for a description of parameters.

2.1.1. Procedures

A node processing a Path message containing a Generalized Label Request must verify that the requested parameters can be satisfied by the interface on which the incoming label is to be allocated, the node itself, and by the interface on which the traffic will be transmitted. The node may either directly support the LSP or it may use a tunnel (FA), i.e., another class of switching. In either case, each parameter must be checked.

Note that local node policy dictates when tunnels may be used and when they may be created. Local policy may allow for tunnels to be dynamically established or may be solely administratively controlled. For more information on tunnels and processing of ER hops when using tunnels see [MPLS-HIERARCHY].

Transit and egress nodes MUST verify that the node itself and, where appropriate, that the interface or tunnel on which the traffic will be transmitted can support the requested LSP Encoding Type. If encoding cannot be supported, the node MUST generate a PathErr message, with a "Routing problem/Unsupported Encoding" indication.

Nodes MUST verify that the type indicated in the Switching Type parameter is supported on the corresponding incoming interface. If the type cannot be supported, the node MUST generate a PathErr message with a "Routing problem/Switching Type" indication.

The G-PID parameter is normally only examined at the egress. If the indicated G-PID cannot be supported then the egress MUST generate a PathErr message, with a "Routing problem/Unsupported L3PID" indication. In the case of PSC and when penultimate hop popping (PHP) is requested, the penultimate hop also examines the (stored) G-PID during the processing of the Resv message. In this case if the G-PID is not supported, then the penultimate hop MUST generate a ResvErr message with a "Routing problem/Unacceptable label value"

indication. The generated ResvErr message MAY include an Acceptable Label Set, see Section 4.1.

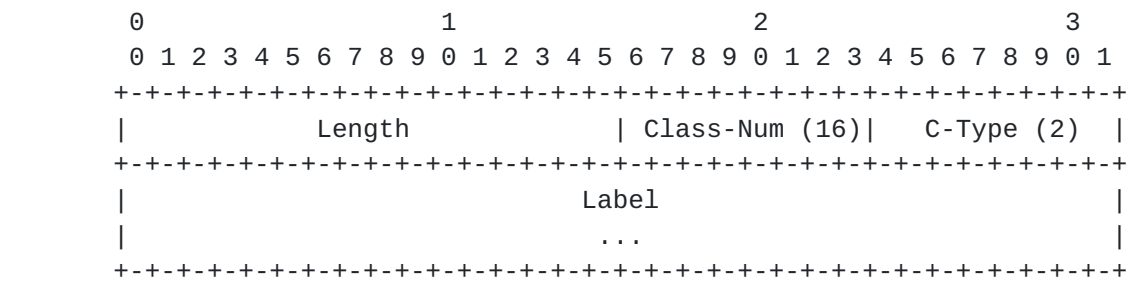
When an error message is not generated, normal processing occurs. In the transit case this will typically result in a Path message being propagated. In the egress case and PHP special case this will typically result in a Resv message being generated.

2.1.2. Bandwidth Encoding

Bandwidth encodings are carried in the SENDER_TSPEC and FLOWSPEC objects. See [GMPLS-SIG] for a definition of values to be used for specific signal types. These values are set in the Peak Data Rate field of Int-Serv objects. Other bandwidth/service related parameters in the object are ignored and carried transparently.

2.2. Generalized Label Object

The format of a Generalized Label object is:



See [GMPLS-SIG] for a description of parameters and encoding of labels.

2.2.1. Procedures

The Generalized Label travels in the upstream direction in Resv messages.

The presence of both a generalized and normal label object in a Resv message is a protocol error and should treated as a malformed message by the recipient.

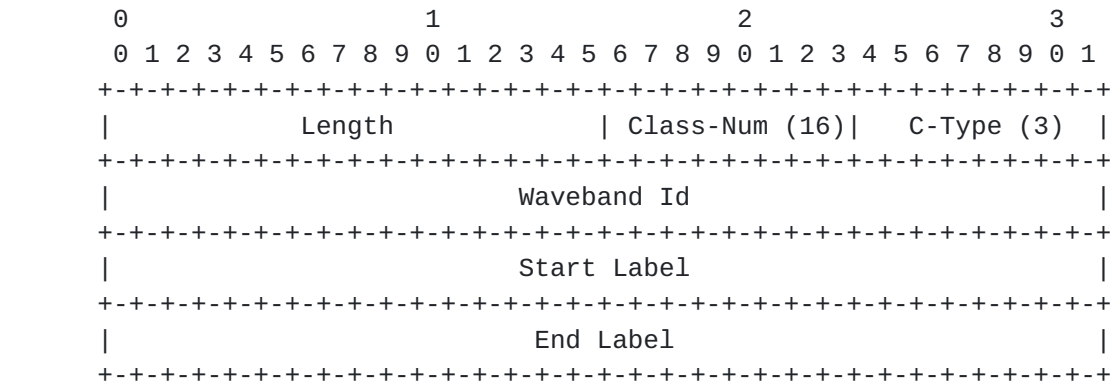
The recipient of a Resv message containing a Generalized Label verifies that the values passed are acceptable. If the label is

unacceptable then the recipient MUST generate a ResvErr message with a "Routing problem/MPLS label allocation failure" indication.

2.3. Waveband Switching

Waveband switching uses the same format as the generalized label, see section 2.2. Waveband Label uses C-Type (3),

In the context of waveband switching, the generalized label has the following format:



See [GMPLS-SIG] for a description of parameters.

2.3.1. Procedures

The procedures defined in Section 2.2.1 apply to waveband switching. This includes generating a ResvErr message with a "Routing problem/MPLS label allocation failure" indication if any of the label fields are unrecognized or unacceptable.

Additionally, when a waveband is switched to another waveband, it is possible that the wavelengths within the waveband will be mirrored about a center frequency. When this type of switching is employed, the start and end label in the waveband label object MUST be flipped before forwarding the label object with the new waveband Id. In this manner an egress/ingress LSR which receives a waveband label which has these values inverted, knows that it must also invert its egress association to pick up the proper wavelengths.

This operation MUST be performed in both directions when a bidirectional waveband tunnel is being established.

2.4. Suggested Label

The format of a Suggested_Label object is identical to a generalized label. It is used in Path messages. A Suggested_Label object uses Class-Number TBA (of form 10bbbbbb) and the C-Type of the label being suggested.

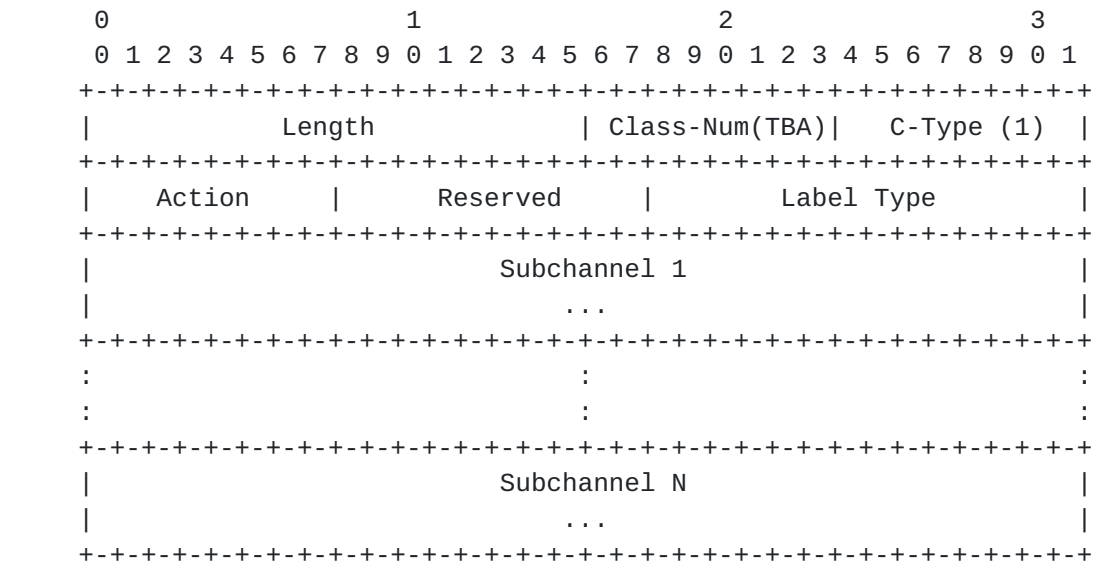
Errors in received Suggested_Label objects MUST be ignored. This includes any received inconsistent or unacceptable values.

Per [GMPLS-SIG], if a downstream node passes a label value that differs from the suggested label upstream, the upstream LSR MUST either reconfigure itself so that it uses the label specified by the downstream node or generate a ResvErr message with a "Routing problem/Unacceptable label value" indication. Furthermore, an ingress node SHOULD NOT transmit data traffic using a suggested label until the downstream node passes a corresponding label upstream.

2.5. Label Set

The Label_Set object uses Class-Number TBA (of form 0bbbbbbb) and the C-Type of 1. It is used in Path messages.

The format of a Label_Set is:



Label Type: 14 bits

Indicates the type and format of the labels carried in the object. Values match the C-Type of the appropriate Label object. Only the low order 8 bits are used in this field.

See [GMPLS-SIG] for a description of other parameters.

2.5.1. Procedures

A Label Set is defined via one or more Label_Set objects. Specific labels/subchannels can be added to or excluded from a Label Set via Action zero (0) and one (1) objects respectively. Ranges of labels/subchannels can be added to or excluded from a Label Set via Action two (2) and three (3) objects respectively. When the Label_Set objects only list labels/subchannels to exclude, this implies that all other labels are acceptable.

The absence of any Label_Set objects implies that all labels are acceptable. A Label Set is included when a node wishes to restrict the label(s) that may be used downstream.

On reception of a Path message, the receiving node will restrict its choice of labels to one which is in the Label Set. Nodes capable of performing label conversion may also remove the Label Set prior to forwarding the Path message. If the node is unable to pick a label from the Label Set or if there is a problem parsing the Label_Set objects, then the request is terminated and a PathErr message with a "Routing problem/Label Set" indication MUST be generated. It is a local matter if the Label Set is stored for later selection on the Resv or if the selection is made immediately for propagation in the Resv.

On reception of a Path message, the Label Set represented in the message is compared against the set of available labels at the downstream interface and the resulting intersecting Label Set is forwarded in a Path message. When the resulting Label Set is empty, the Path must be terminated, and a PathErr message, and a "Routing problem/Label Set" indication MUST be generated. Note that intersection is based on the physical labels (actual wavelength/band values) which may have different logical values on different links, as a result it is the responsibility of the node to map these values so that they have a consistent physical meaning, or to drop the particular values from the set if no suitable logical label value exists.

When processing a Resv message at an intermediate node, the label

propagated upstream MUST fall within the Label Set.

Note, on reception of a Resv message a node that is incapable of performing label conversion has no other choice than to use the same physical label (wavelength/band) as received in the Resv message. In this case, the use and propagation of a Label Set will significantly reduce the chances that this allocation will fail.

3. Bidirectional LSPs

Bidirectional LSP setup is indicated by the presence of an Upstream Label in the Path message. An Upstream_Label object has the same format as the generalized label, see Section 2.2. The Upstream_Label object uses Class-Number TBA (of form 0bbbbbbb) and the C-Type of the label being used.

3.1. Procedures

The process of establishing a bidirectional LSP follows the establishment of a unidirectional LSP with some additions. To support bidirectional LSPs an Upstream_Label object is added to the Path message. The Upstream_Label object MUST indicate a label that is valid for forwarding at the time the Path message is sent.

When a Path message containing an Upstream_Label object is received, the receiver first verifies that the upstream label is acceptable. If the label is not acceptable, the receiver MUST issue a PathErr message with a "Routing problem/Unacceptable label value" indication. The generated PathErr message MAY include an Acceptable Label Set, see Section 4.1.

An intermediate node must also allocate a label on the outgoing interface and establish internal data paths before filling in an outgoing upstream label and propagating the Path message. If an intermediate node is unable to allocate a label or internal resources, then it MUST issue a PathErr message with a "Routing problem/MPLS label allocation failure" indication.

Terminator nodes process Path messages as usual, with the exception that the upstream label can immediately be used to transport data traffic associated with the LSP upstream towards the initiator.

When a bidirectional LSP is removed, both upstream and downstream labels are invalidated and it is no longer valid to send data using the associated labels.

3.2. Contention Resolution

There are two additional contention resolution related considerations when controlling bidirectional LSP setup via RSVP-TE. The first is that for the purposes of RSVP contention resolution, the node ID is the IP address used in the RSVP_HOP object. The second is that a neighbor's node ID might not be known when sending an initial Path message. When this case occurs, a node should suggest a label chosen at random from the available label space.

4. Notification

This section covers several notification related extensions. The first extension defines the Acceptable Label Set object to support Notification on Label Error, per [GMPLS-SIG]. The second and third extensions enable expedited notification of failures and other events to nodes responsible for restoring failed LSPs. (The second extension, the Notify Request object, identifies where event notifications are to be sent. The third extension, the Notify message, provides for general event notification.) The final notification related extension allows for the removal of Path state on handling of PathErr messages.

4.1. Acceptable Label Set Object

Acceptable_Label_Set objects use a Class-Number TBA (of form 10bbbbbb). The remaining contents of the object, including C-Type, have the identical format as the Label_Set object, see Section 2.5.

Acceptable_Label_Set objects may be carried in PathErr and ResvErr messages. The procedures for defining an Acceptable Label Set follow the procedures for defining a Label Set, see Section 2.5.1. Specifically, an Acceptable Label Set is defined via one or more Acceptable_Label_Set objects. Specific labels/subchannels can be added to or excluded from an Acceptable Label Set via Action zero (0) and one (1) objects respectively. Ranges of labels/subchannels can be added to or excluded from an Acceptable Label Set via Action two (2) and three (3) objects respectively. When the Acceptable_Label_Set objects only list labels/subchannels to exclude, this implies that all other labels are acceptable.

The inclusion of Acceptable_Label_Set objects is optional. If included, the PathErr or ResvErr message SHOULD contain a "Routing problem/Unacceptable label value" indication. The absence of Acceptable_Label_Set objects does not have any specific meaning.

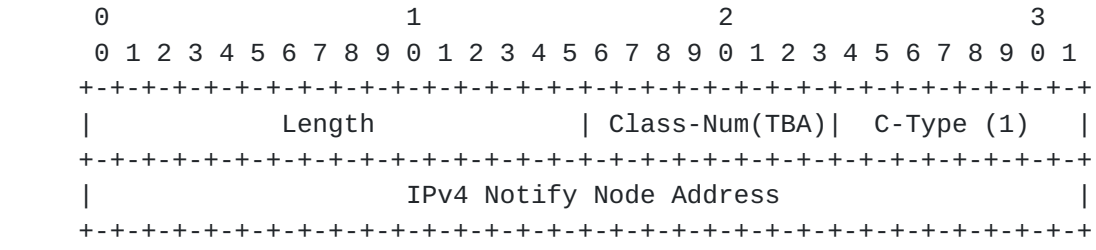
4.2. Notify Request Objects

Notifications may be sent via the Notify message defined below. The Notify Request object is used to request the generation of notifications. Notifications, i.e., the sending of a Notify message, may be requested in both the upstream and downstream directions.

4.2.1. Required Information

The Notify Request Object may be carried in Path or Resv Messages, see Section 7. The Notify_Request Class-Number is TBA (of form 11bbbbbb). The format of a Notify Request is:

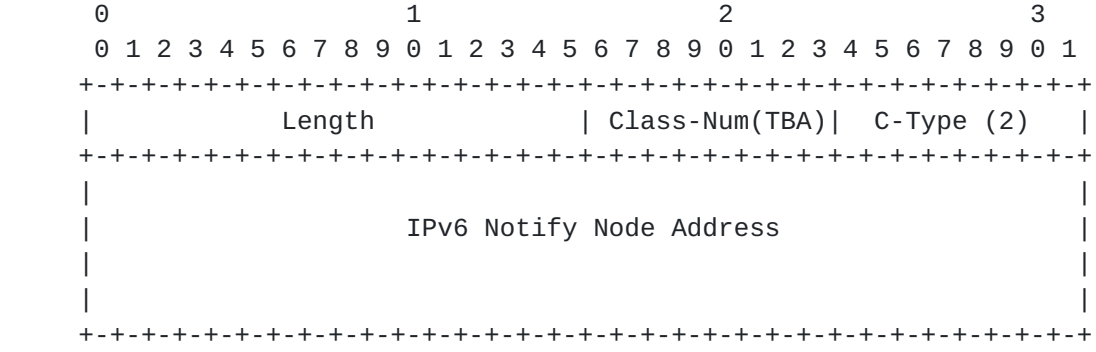
o IPv4 Notify Request Object



IPv4 Notify Node Address: 32 bits

The IP address of the node that should be notified when generating an error message.

o IPv6 Notify Request Object



IPv6 Notify Node Address: 16 bytes

The IP address of the node that should be notified when generating an error message.

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.

4.2.2. Procedures

A Notify Request object may be inserted in Path or Resv messages to indicate the address of a node that should be notified of an LSP failure. As previously mentioned, notifications may be requested in both the upstream and downstream directions. Upstream notification is indicated via the inclusion of a Notify Request Object in the corresponding Path message. Downstream notification is indicated via the inclusion of a Notify Request Object in the corresponding Resv message.

A node receiving a message containing a Notify Request object SHOULD store the Notify Node Address in the corresponding state block. If the node is a transit node, it SHOULD also included a Notify Request object in the outgoing Path or Resv message. The outgoing Notify Node Address MAY be updated based on local policy.

Note that the inclusion of a Notify Request object does not guarantee that a Notify message will be generated.

4.3. Notify Message

The Notify message provides a mechanism to inform non-adjacent nodes of LSP related events. Notify messages are normally generated only after a Notify Request object has been received. The Notify message differs from the currently defined error messages (i.e., PathErr and ResvErr messages) in that it can be "targeted" to a node other than the immediate upstream or downstream neighbor and that it is a generalized notification mechanism. The Notify message does not replace existing error messages. The Notify message may be sent either (a) normally, where non-target nodes just forward the Notify message to the target node, similar to ResvConf processing in [RSVP]; or (b) encapsulated in a new IP header whose destination is equal to the target IP address. Regardless of the transmission mechanism, nodes receiving a Notify message not destined to the node forward the message, unmodified, towards the target.

To support reliable delivery of the Notify message, an Ack Message [RFC2961] is used to acknowledge the receipt of a Notify Message. See [RFC2961] for details on reliable RSVP message delivery.

4.3.1. Required Information

The Notify message is a generalized notification message. The IP destination address is set to the IP address of the intended receiver. The Notify message is sent without the router alert option. A single Notify message may contain notifications being sent, with respect to each listed session, both upstream and downstream.

The Notify message has a Message Type of TBA (by IANA). The Notify message format is as follows:

```
<Notify message>            ::= <Common Header> [<INTEGRITY>]
                      [ [<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ... ]
                                [ <MESSAGE_ID> ]
                                <ERROR_SPEC> <notify session list>

<notify session list>       ::= [ <notify session list> ]
                                <upstream notify session> |
                                <downstream notify session>

<upstream notify session>   ::= <SESSION> [ <ADMIN_STATUS> ]
                                [<POLICY_DATA>...]
                                <sender descriptor>

<downstream notify session> ::= <SESSION> [<POLICY_DATA>...]
                                <flow descriptor list>
```

The ERROR_SPEC object specifies the error and includes the IP address of either the node that detected the error or the link that has failed. See ERROR_SPEC definition in [RFC2205]. The MESSAGE_ID and related objects are defined in [RFC2961] and are used when refresh reductions is supported.

4.3.2. Procedures

Notify messages are most commonly generated at nodes that detect an error that will trigger the generation of a PathErr or ResvErr message. If a PathErr message is to be generated and a Notify Request object has been received in the corresponding Path message, then a Notify message destined to the recorded node SHOULD be generated. If a ResvErr message is to be generated and a Notify Request object has been received in the corresponding Resv message, then a Notify message destined to the recorded node SHOULD be generated. As previously mentioned, a single error may generate a Notify message in both the upstream and downstream directions. Note that a Notify message MUST NOT be generated unless an appropriate

Notify Request object has been received.

When generating Notify messages, a node SHOULD attempt to combine notifications being sent to the same Notify Node and that share the same ERROR_SPEC into a single Notify message. The means by which a node determines which information may be combined is implementation dependent. Implementations may use event, timer based or other approaches. If using a timer based approach, the implementation SHOULD allow the user to configure the interval over which notifications are combined. When using a timer based approach, a default "notification interval" of 1 ms SHOULD be used. Notify messages SHOULD be delivered using the reliable message delivery mechanisms defined in [RFC2961].

Upon receiving a Notify message, the Notify Node SHOULD send a corresponding Ack message.

4.4. Removing State with a PathErr message

The PathErr message as defined in [RFC2205] is sent hop-by-hop to the source of the associated Path message. Intermediate nodes may inspect this message, but take no action upon it. In an environment where Path messages are routed according to an IGP and that route may change dynamically, this behavior is a fine design choice.

However, when RSVP is used with explicit routes, it is often the case that errors can only be corrected at the source node or some other node further upstream. In order to clean up resources, the source must receive the PathErr and then either send a PathTear (or wait for the messages to timeout). This causes idle resources to be held longer than necessary and increases control message load. In a situation where the control plane is attempting to recover from a serious outage, both the message load and the delay in freeing resources hamper the ability to rapidly reconverge.

The situation can be greatly improved by allowing state to be removed by intermediate nodes on certain error conditions. To facilitate this a new flag is defined in the ERROR_SPEC object. The two currently defined ERROR_SPEC objects (IPv4 and IPv6 error spec objects) each contain a one byte flag field. Within that field two flags are defined. This specification defines a third flag, 0x04, Path_State_Removed.

The semantics of the Path_State_Removed flag are simply that the node forwarding the error message has removed the Path state associated with the PathErr. By default, the Path_State_Removed flag is always set to zero when generating or forwarding a PathErr message. A node

which encounters an error MAY set this flag if the error results in the associated Path state being discarded. If the node setting the flag is not the session endpoint, the node SHOULD generate a corresponding PathTear. A node receiving a PathErr message containing an ERROR_SPEC object with the Path_State_Removed flag set MAY also remove the associated Path state. If the Path state is removed the Path_State_Removed flag SHOULD be set in the outgoing PathErr message. A node which does not remove the associated Path state MUST NOT set the Path_State_Removed flag. A node that receives an error with the Path_State_Removed flag set to zero MUST NOT set this flag unless it also generates a corresponding PathTear message.

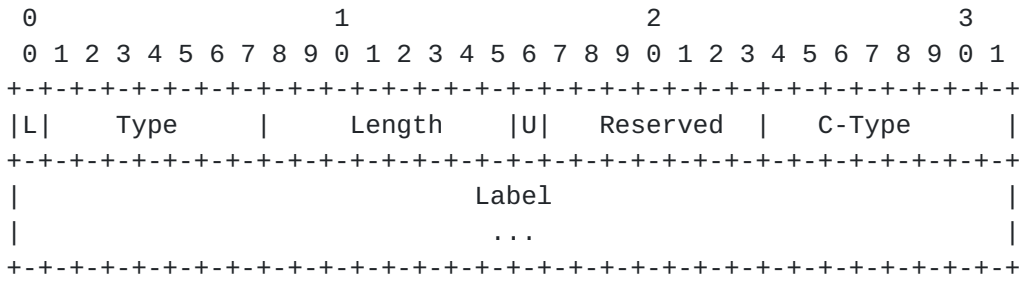
Note that the use of this flag does not result in any interoperability incompatibilities.

5. Explicit Label Control

The Label ERO and RRO subobjects are defined to support Explicit Label Control. Note that the Label RRO subobject was defined in [RFC3209] and is being revised to support bidirectional LSPs.

5.1. Label ERO subobject

The Label ERO subobject is defined as follows:



See [GMPLS-SIG] for a description of L, U and Label parameters.

Type

3 Label

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always divisible by 4.

C-Type

The C-Type of the included Label Object. Copied from the Label Object.

5.1.1. Procedures

The Label subobject follows a subobject containing the IP address, or the interface identifier [MPLS-UNNUM], associated with the link on which it is to be used. Up to two label subobjects may be present, one for the downstream label and one for the upstream label. The following SHOULD result in "Bad EXPLICIT_ROUTE object" errors:

- If the first label subobject is not preceded by a subobject containing an IP address, or a interface identifier [MPLS-UNNUM], associated with an output link.
- For a label subobject to follow a subobject that has the L-bit set
- On unidirectional LSP setup, for there to be a label subobject with the U-bit set
- For there to be two label subobjects with the same U-bit values

To support the label subobject, a node must check to see if the subobject following its associate address/interface is a label subobject. If it is, one subobject is examined for unidirectional LSPs and two subobjects for bidirectional LSPs. If the U-bit of the subobject being examined is clear (0), then value of the label is copied into a new Label_Set object. This Label_Set object MUST be included on the corresponding outgoing Path message.

If the U-bit of the subobject being examined is set (1), then value of the label is label to be used for upstream traffic associated with the bidirectional LSP. If this label is not acceptable, a "Bad EXPLICIT_ROUTE object" error SHOULD be generated. If the label is acceptable, the label is copied into a new Upstream_Label object. This Upstream_Label object MUST be included on the corresponding outgoing Path message.

After processing, the label subobjects are removed from the ERO.

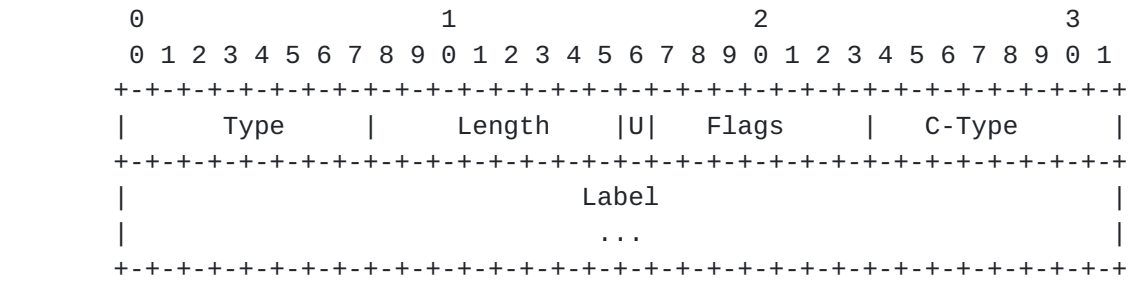
Note an implication of the above procedures is that the label subobject should never be the first subobject in a newly received

message. If the label subobject is the the first subobject an a received ERO, then it SHOULD be treated as a "Bad strict node" error.

Procedures by which an LSR at the head-end of an LSP obtains the information needed to construct the Label subobject are outside the scope of this document.

5.2. Label RRO subobject

The Label RRO subobject is defined as follows:



See [GMPLS-SIG] for a description of U and Label parameters.

Type

3 Label

Length

See [RFC3209].

Flags

See [RFC3209].

C-Type

The C-Type of the included Label Object. Copied from the Label Object.

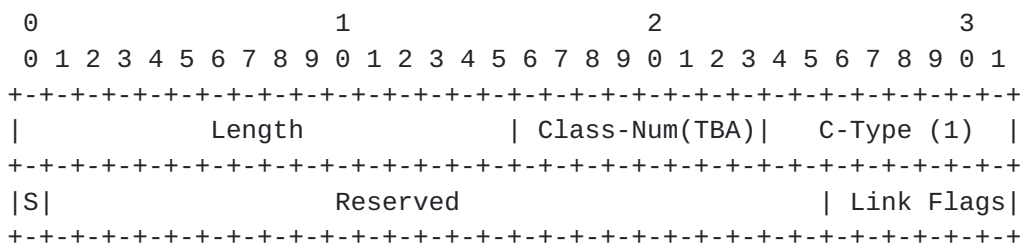
5.2.1. Procedures

Label RRO subobjects are included in RROs as described in [RFC3209]. The only modification to usage and processing from [RFC3209] is that when labels are recorded for bidirectional LSPs, label ERO subobjects for both downstream and upstream labels MUST be included.

6. Protection Object

The use of the Protection Object is optional. The object is included to indicate specific protection attributes of an LSP. The Protection Object uses Class-Number TBA (of form 0bbbbbbb).

The format of the Protection Object is:



See [GMPLS-SIG] for a description of parameters.

6.1. Procedures

Transit nodes processing a Path message containing a Protection Object MUST verify that the requested protection can be satisfied by the outgoing interface or tunnel (FA). If it cannot, the node MUST generate a PathErr message, with a "Routing problem/Unsupported Link Protection" indication.

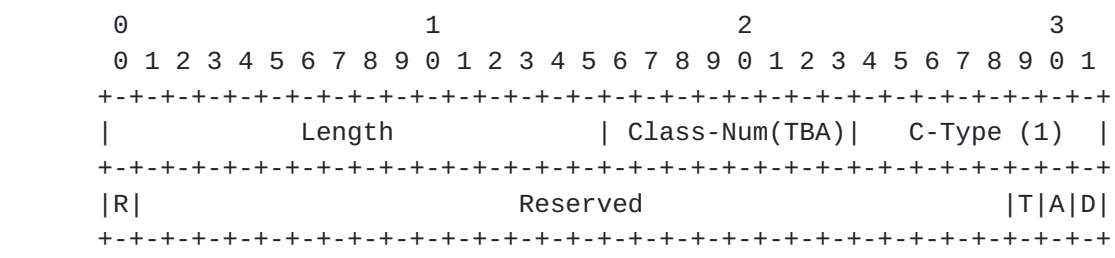
7. Administrative Status Information

Administrative Status Information is carried in the Admin_Status object. The object provides information related to the administrative state of a particular LSP. The information is used in two ways. In the first, the object is carried in Path and Resv messages to indicate the administrative state of an LSP. In the second, the object is carried in a Notification message to request that the ingress node change the administrative state of an LSP.

7.1. Admin Status Object

The use of the Admin_Status Object is optional. It uses Class-Number TBA (of form 11bbbbbb).

The format of the Admin_Status Object is:



See [GMPLS-SIG] for a description of parameters.

7.2. Path and Resv Message Procedures

The Admin_Status object is used to notify each node along the path of the status of the LSP. Status information is processed by each node based on local policy and then propagated in the corresponding outgoing messages. The object may be inserted in either Path or Resv messages at the discretion of the ingress (for Path messages) or egress (for Resv messages) nodes. The absence of the object is equivalent to receiving an object containing values all set to zero (0).

Transit nodes receiving a non-refresh Path or Resv message containing an Admin_Status object, update their local state, take any appropriate local action based on the indicated status and then propagate the received Admin_Status object in the corresponding outgoing Path or Resv message. If the values of an Admin_Status object received in a Resv message differs from the values received in a Path message then, with one exception, no local action should be taken but the values should still be propagated. The one case where values received in the Resv message should result in local action is when both the received R and D bits are set, i.e., are one (1).

Edge nodes receiving a non-refresh Path or Resv message containing an Admin_Status object, also update their local state and take any appropriate local action based on the indicated status. When an Admin Status object is received with the R bit set, the receiving edge node should reflect the received values in a corresponding outgoing message. Specifically, if an egress node receives a Path message with the R bit of the Admin_Status object set and the node has previously issued a Resv message corresponding to the Path message, the node SHOULD send an updated Resv message containing an Admin_Status object with the same values set, with the exception of the R bit, as received in the corresponding Path message. Furthermore, the egress node SHOULD also ensure that subsequent Resv messages sent by the node contain the same Admin Status Object.

Additionally, if an ingress node receives a Resv message with the R bit of the Admin_Status object set, the node SHOULD send an updated Path message containing an Admin_Status object with the same values set, with the exception of the R bit, as received in the corresponding Resv message. Furthermore, the ingress node SHOULD also ensure that subsequent Path messages sent by the node contain the same Admin Status Object.

7.2.1. Deletion procedure

In some circumstances, particularly optical networks, it is useful to set the administrative status of an LSP before tearing it down. In such circumstances the procedure SHOULD be followed when deleting an LSP from the ingress:

1. The ingress node precedes an LSP deletion by inserting an Admin Status Object in a Path message and setting the Reflect (R) and Delete (D) bits.

2. Transit and egress nodes process the Admin Status Object as described above. (Alternatively, the egress MAY respond with a PathErr message with the Path_State_Removed flag set, see section 4.4.)

3. Upon receiving the Admin Status Object with the Delete (D) bit set in the Resv message, the ingress node sends a PathTear message downstream to remove the LSP and normal RSVP processing takes place.

In such circumstances the procedure SHOULD be followed when deleting an LSP from the egress:

1. The egress node indicates its desire for deletion by inserting an Admin Status Object in a Resv message and setting the Reflect (R) and Delete (D) bits.

2. Transit nodes process the Admin Status Object as described above.

3. Upon receiving the Admin Status Object with the Delete (D) bit set in the Resv message, the ingress node sends a PathTear message downstream to remove the LSP and normal RSVP processing takes place.

7.2.2. Compatibility and Error Procedures

It is possible that some nodes along an LSP will not support the Admin Status Object. In the case of a non-supporting transit node, the object will pass through the node unmodified and normal processing can continue. In the case of a non-supporting egress node, the Admin Status Object will not be reflected back in the Resv Message. To support the case of a non-supporting egress node, the ingress SHOULD only wait a configurable period of time for the updated Admin Status Object in a Resv message. Once the period of time has elapsed, the ingress node sends a PathTear message. By default this period of time SHOULD be 30 seconds.

7.3. Notify Message Procedures

Intermediate and egress nodes may trigger the setting of administrative status via the use of Notify messages. To accomplish this, an intermediate or egress node generates a Notify message with the corresponding upstream notify session information. The Admin Status Object MUST be included in the session information, with the appropriate bit or bits set. The Reflect (R) bit MUST NOT be set. The Notify message may be, but is not required to be, encapsulated, see Section 4.3.

An ingress node receiving a Notify message containing an Admin Status Object with the Delete (D) bit set, SHOULD initiate the deletion procedure described in the previous section. Other bits SHOULD be propagated in an outgoing Path message as normal.

7.3.1. Compatibility and Error Procedures

Some special processing is required in order to cover the case of nodes that do not support the Admin Status Object and other error conditions. Specifically, a node that sends a Notify message containing an Admin Status Object with the Down (D) bit set MUST verify that it receives a corresponding Path message with the Down (D) bit set within a configurable period of time. By default this period of time SHOULD be 30 seconds. If the node does not receive such a Path message, it SHOULD send a PathTear message downstream and either a ResvTear message or a PathErr message with the Path_State_Removed flag set upstream.

8. Control Channel Separation

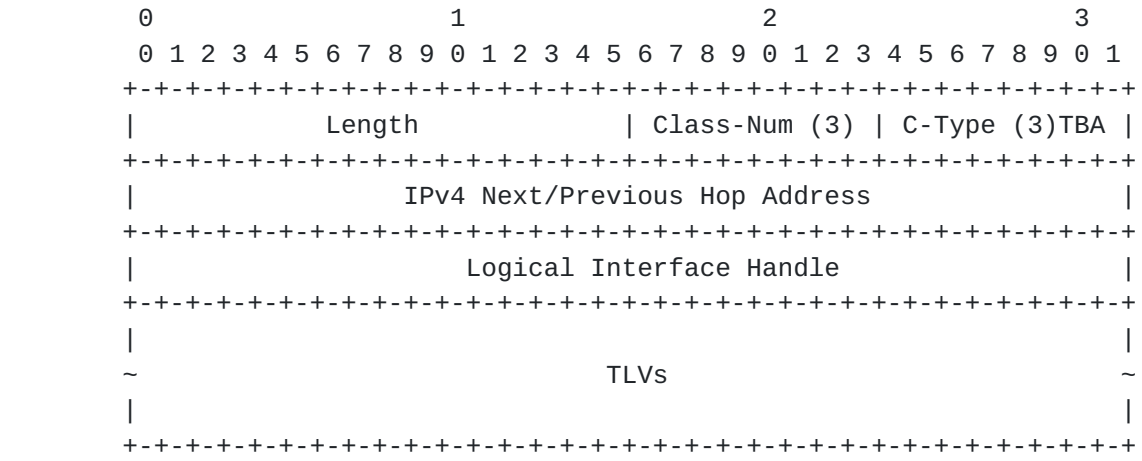
This section provides the protocol specific formats and procedures to required support a control channel not being in-band with a data channel.

8.1. Interface Identification

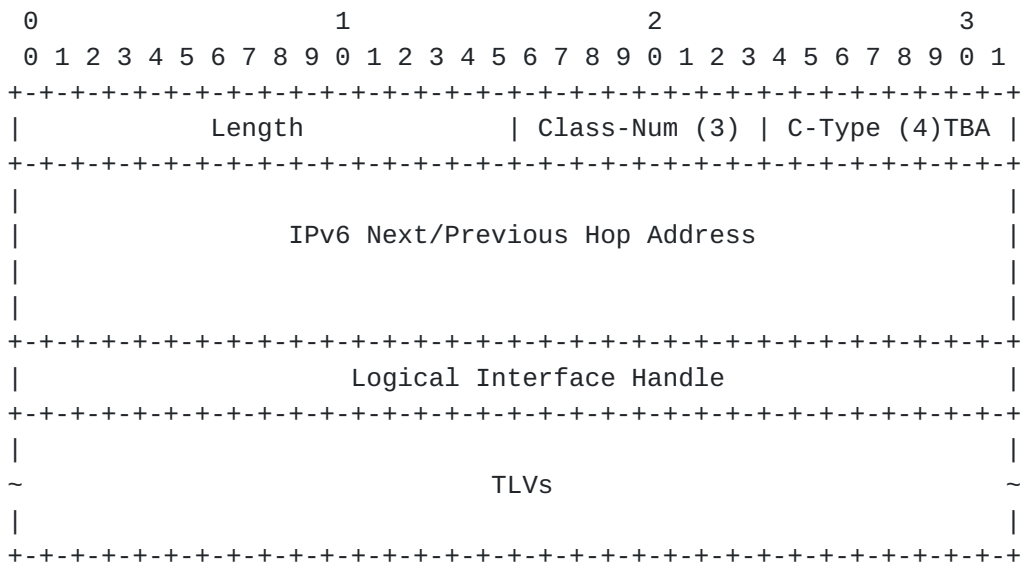
The choice of the data interface to use is always made by the sender of the Path message. The choice of the data interface is indicated by the sender of the Path message by including the data channel's interface identifier in the message using a new RSVP_HOP object sub-type. For bidirectional LSPs, the sender chooses the data interface in each direction. In all cases but bundling, the upstream interface is implied by the downstream interface. For bundling, the path sender explicitly identifies the component interface used in each direction. The new RSVP_HOP object is used in Resv message to indicate the downstream node's usage of the indicated interface(s).

8.1.1. IF_ID RSVP_HOP Objects

The format of the IPv4 IF_ID RSVP_HOP Object is:



The format of the IPv6 IF_ID RSVP_HOP Object is:



See [RFC2205] for a description of hop address and handle fields. See [GMPLS-SIG] for a description of parameters and encoding of TLVs.

8.1.2. Procedures

An IF_ID RSVP_HOP object is used in place of previously defined RSVP_HOP objects. It is used on links where there is not a one-to-one association of a control channel to a data channel, see [GMPLS-SIG]. The Hop Address and Logical Interface Handle fields are used per standard RSVP [RFC2205].

TLVs are used to identify the data channel(s) associated with an LSP. For a unidirectional LSP, a downstream data channel MUST be indicated. For bidirectional LSPs, a common downstream and upstream data channel is normally indicated. In the special case where a bidirectional LSP that traverses a bundled link, it is possible to specify a downstream data channel that differs from the upstream data channel. Data channels are specified from the view point of the sender of the Path message. The IF_ID RSVP_HOP object SHOULD NOT be used when no TLVs are needed.

A node receiving one or more TLVs in a Path message saves their values and returns them in the HOP objects of subsequent Resv messages sent to the node that originated the TLVs.

As with [MPLS-TE], the node originating an IF_ID object must ensure that the selected outgoing interface is consistent with the outgoing

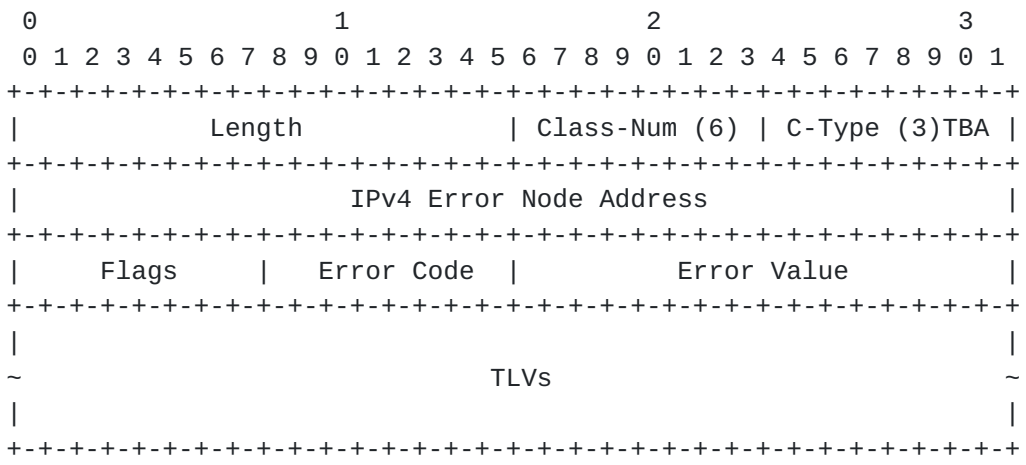
ERO. A node that receives an IF_ID object SHOULD check whether the information carried in this object is consistent with the information carried in a received ERO, and if not it MUST send a PathErr with the error code "Routing Error" and error value of "Bad Explicit Route Object" toward the sender.

8.2. Errored Interface Identification

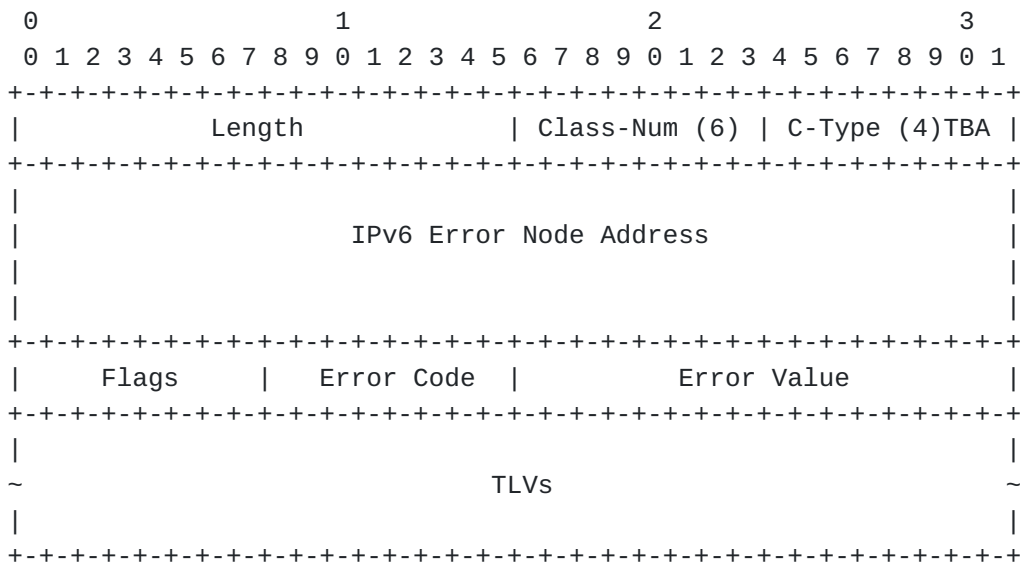
There are cases where it is useful to indicate a specific interface associated with an error. To support these cases the IF_ID ERROR_SPEC Objects are defined.

8.2.1. IF_ID ERROR_SPEC Objects

The format of the IPv4 IF_ID ERROR_SPEC Object is:



The format of the IPv6 IF_ID ERROR_SPEC Object is:



See [RFC2205] for a description of address, flags, error code and error value fields. See [GMPLS-SIG] for a description of parameters and encoding of TLVs.n

8.2.2. Procedures

Nodes wishing to indicate that an error is related to a specific interface SHOULD use the appropriate IF_ID ERROR_SPEC Object in the corresponding PathErr or ResvErr message. IF_ID ERROR_SPEC Objects SHOULD be generated and processed as any other ERROR_SPEC Object, see [RFC2205].

9. Fault Handling

The handling of two types of control communication faults is described in this section. The first, referred to as nodal faults, relates to the case where a node losses its control state (e.g., after a restart) but does not loose its data forwarding state. In the second, referred to as control channel faults, relates to the case where control communication is lost between two nodes. The handling of both faults is supported by the Restart_Cap object defined below and require the use of Hello messages.

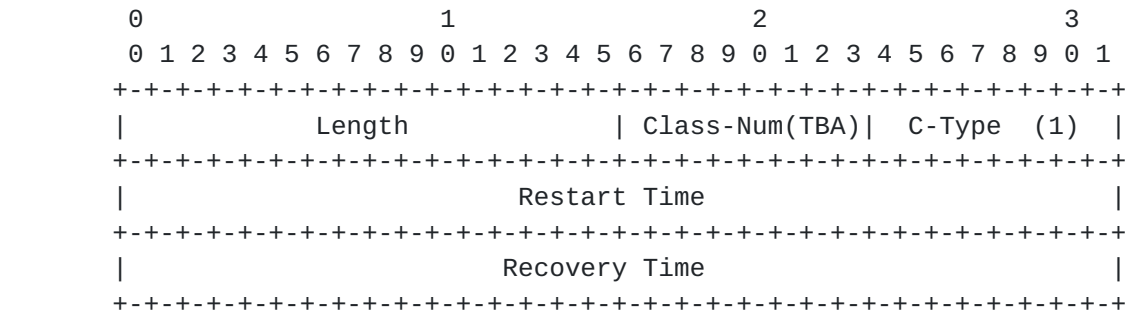
Note, the Restart_Cap object MUST NOT be sent when there is no mechanism to detect data channel failures independent of control channel failures.

Please note this section is derived from [PAN-RESTART].

9.1. Restart_Cap Object

The Restart_Cap Object is carried in Hello messages.

The format of the Restart_Cap Object is:



Restart Time: 32 bits

Restart Time is measured in milliseconds. Restart Time SHOULD be set to the sum of the time it takes the sender of the object to restart its RSVP-TE component (to the point where it can exchange RSVP Hello with its neighbors) and the communication channel that is used for RSVP communication. A value of 0xffffffff indicates that the restart of the sender's control plane may occur over an indeterminate interval and that the operation of its data plane is unaffected by control plane failures. The method used to ensure continued data plane operation is outside the scope of this document.

Recovery Time: 32 bits

The period of time, in milliseconds, that the sender desires for the recipient to re-synchronize RSVP and MPLS forwarding state with the sender after the re-establishment of Hello synchronization. A value of zero (0) indicates that MPLS forwarding state was not preserved across a particular reboot.

9.2. Processing of Restart_Cap Object

Nodes supporting state recovery advertise this capability by carrying the Restart_Cap object in Hello messages. Such nodes MUST include the Restart_Cap object in all Hello messages. (Note that this includes Hello messages containing ACK objects.) Usage of the special case Recovery Time values is described in greater detail below.

When a node receives a Hello message with the Restart_Cap object, it SHOULD record the values of the parameters received.

9.3. Modification to Hello Processing to Support State Recovery

When a node determines that RSVP communication with a neighbor has been lost, and the node previously learned that the neighbor supports state recovery, the node SHOULD wait at least the amount of time indicated by the Restart Time indicated by the neighbor before invoking procedures related to communication loss. A node MAY wait longer based on local policy or configuration information.

During this waiting period, all Hello messages MUST be sent with a Dst_Instance value set to zero (0), and Src_Instance should be unchanged. While waiting, the node SHOULD also preserve the RSVP and MPLS forwarding state for (already) established LSPs that traverse the link(s) between the node and the neighbor. In a sense with respect to established LSPs the node behaves as if it continues to receive periodic RSVP refresh messages from the neighbor. The node MAY clear RSVP and forwarding state for the LSPs that are in the process of being established when their refresh timers expire. Refreshing of Resv and Path state SHOULD be suppressed during this waiting period.

During this waiting period, the node MAY inform upstream nodes of the communication loss via a PathErr and/or upstream Notify message with "Control Channel Degraded State" indication. If such notification has been sent, then upon restoration of the control channel the node MUST inform other nodes of the restoration via a PathErr and/or upstream Notify message with "Control Channel Active State" indication. (Specific error codes are to be assigned IANA.)

When a new Hello message is received from the neighbor, the node must determine if the fault was limited to the control channel or was a nodal fault. This determination is based on the Src_Instance received from the neighbor. If the value is different than the value that was received from the neighbor prior to the fault, then the neighbor should be treated as if it has restarted. Otherwise, the

the fault was limited control channel. Procedures for handling each case are described below.

9.4. Control Channel Faults

In the case of control channel faults, the node SHOULD refresh all state shared with the neighbor. Summary Refreshes [RFC2961] with the ACK_Desired flag set SHOULD be used, if supported. Note that if a large number of messages are need, some pacing should be applied. All state SHOULD be refreshed within the Recovery time advertised by the neighbor.

9.5. Nodal Faults

Recovering from nodal faults uses one new object and other existing protocol messages and objects.

9.5.1. Recovery Label

The Recovery_Label object is used during the nodal fault recovery process. The format of a Recovery_Label object is identical to a generalized label. A Recovery_Label object uses Class-Number TBA (of form 0bbbbbbb) and the C-Type of the label being suggested.

9.5.2. Procedures for the Restarting node

After a node restarts its control plane, a node that supports state recovery SHOULD check whether it was able to preserve its MPLS forwarding state. If no forwarding state from prior to the restart was preserved, then the node MUST set the Recovery Time to 0 in the Hello message the node sends to its neighbors.

If the forwarding state was preserved, then the node initiates the state recovery process. The period during which a node is prepared to support the recovery process is referred to as the Recovery Period. The total duration of the Recovery Period is advertised by the recovering node in the Recovery Time parameter of the Restart_Cap object. The Recovery Time MUST be set to the duration of the Recovery Period in all Hello messages sent during the Recovery Period. State that is not resynchronized during the Recovery Period SHOULD be removed at the end of the Period.

Note that if during Hello synchronization the restarting node

determines that a neighbor does not support state recovery, and the restarting node maintains its MPLS forwarding state on a per neighbor basis, the restarting node should immediately consider the Recovery Period with that neighbor completed. Forwarding state may be considered to be maintained on a per neighbor basis when per interface labels are used on point-to-point interfaces.

When a node receives a Path message during the Recovery Period, the node first checks if it has an RSVP state associated with the message. If the state is found, then the node handles this message according to previously defined procedures.

If the RSVP state is not found, and the message does not carry a Recovery_Label object, the node treats this as a setup for a new LSP, and handles it according to previously defined procedures.

If the RSVP state is not found, and the message carries a Recovery_Label object, the node searches its MPLS forwarding table (the one that was preserved across the restart) for an entry whose incoming interface matches the Path message and whose incoming label is equal to the label carried in the Recovery_Label object.

If the MPLS forwarding table entry is not found, the node treats this as a setup for a new LSP, and handles it according to previously defined procedures.

If the MPLS forwarding table entry is found, the appropriate RSVP state is created, the entry is bound to the LSP associated with the message, and related forwarding state should be considered as valid and refreshed. Normal Path message processing should also be conducted. When sending the corresponding outgoing Path message the node SHOULD include a Suggested_Label object with a label value matching the outgoing label from the now restored forwarding entry. The outgoing interface SHOULD also be selected based on the forwarding entry. In the special case where a restarting node also has a restating downstream neighbor, a Recovery_Label object should be used instead of a Suggested_Label object.

Additionally, for bidirectional LSPs, the node extracts the label from the UPSTREAM_LABEL object carried in the received Path message, and searches its MPLS forwarding table for an entry whose outgoing label is equal to the label carried in the object (in the case of link bundling, this may also involved first identifying the appropriate incoming component link).

If the MPLS forwarding table entry is not found, the node treats this as a setup for a new LSP, and handles it according to previously defined procedures.

If the MPLS forwarding table entry is found, the entry is bound to the LSP associated with the Path message, and the entry should be considered to be re-synchronized. In addition, if the node is not the tail-end of the LSP, the corresponding outgoing Path messages is sent with the incoming label from that entry carried in the UPSTREAM_LABEL object.

During the Recovery Period, Resv messages are processed normally with two exceptions. In the case that a forwarding entry is recovered, no new label or resource allocation is required while processing the Resv message. The second exception is that ResvErr messages SHOULD NOT be generated when a Resv message with no matching Path state is received. In this case the Resv message SHOULD just be silently discarded.

9.5.3. Procedures for the Neighbor of a Restarting node

The following specifies the procedures that apply when the node reestablishes communication with the neighbor's control plane within the Restart Time, the node determines (using the procedures defined in Section 5 of [RFC3209]) that the neighbor's control plane has restarted, and the neighbor was able to preserve its forwarding state across the restart (as was indicated by a non-zero Recovery Time carried in the Restart_Cap object of the RSVP Hello messages received from the neighbor). Note, a Restart Time value of 0xffffffff indicates an infinite Restart Time interval.

Upon detecting a restart with a neighbor that supports state recovery, a node SHOULD refresh all Path state shared with that neighbor. The outgoing Path messages MUST include the Recovery_Label object containing the label value received in the most recently received corresponding Resv message. All Path state SHOULD be refreshed within approximately 1/2 of the Recovery time advertised by the restarted neighbor. If there are many LSP's going through the restarting node, the neighbor node should avoid sending Path messages in a short time interval, as to avoid unnecessary stressing the restarting node's CPU. Instead, it should spread the messages across 1/2 the Recovery Time interval.

After detecting a restart of a neighbor that supports state recovery, all Resv state shared with the restarting node MUST NOT be refreshed until a corresponding Path message is received. This requires suppression of normal Resv and Summary Refresh processing to the neighbor during the Recovery Time advertised by the restarted neighbor. As soon as a corresponding Path message is received a Resv message SHOULD be generated and normal state processing SHOULD be re-enabled.

10. RSVP Message Formats and Handling

This message summarizes RSVP message formats and handling as modified by GMPLS.

10.1. 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. Unmodified formats are not listed. Again, MESSAGE_ID and related objects are defined in [RFC2961].

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> ]
                         [ <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> ]
```

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>
```

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> ... ]
                         [ <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> is not modified by this document.

The format of a ResvErr message is as follows:

```
<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> <error flow descriptor>
```

The modified Hello message format is:

```
<Hello Message> ::= <Common Header> [ <INTEGRITY> ] <HELLO>
                    [ <RESTART_CAP> ]
```

10.2. Addressing Path and PathTear Messages

RSVP was designed to handle dynamic (non-explicit) path changes and non RSVP hops along the path. To this end, the Path and PathTear messages carry the destination address of the session in the IP header. In generalized signaling, routes are usually explicitly signaled. Further, hops that cannot allocate labels cannot exist in the path of an LSP. A further difference with traditional RSVP is that at times, an RSVP message may travel out of band with respect to an LSP's data channel.

When a node is sending a Path or PathTear message to a node that it knows to be adjacent at the data plane (i.e. along the path of the LSP) it SHOULD address the message directly to an address associated with the adjacent node's control plane. In this case the router-alert option SHOULD not be included.

11. Acknowledgments

This draft is the work of numerous authors and consists of a composition of a number of previous drafts in this area. A list of the drafts from which material and ideas were incorporated follows:

draft-saha-rsvp-optical-signaling-00.txt
draft-lang-mpls-rsvp-oxc-00.txt
draft-kompella-mpls-optical-00.txt
draft-fan-mpls-lambda-signaling-00.txt
draft-pan-rsvp-te-restart-01.txt

Valuable comments and input were received from a number of people, including Igor Bryskin, Adrian Farrel and Dimitrios Pendarakis. Portions of Section 4 are based on suggestions and text proposed by Adrian Farrel.

12. Security Considerations

The transmission of notify messages using IP in IP, breaks RSVP's hop-by-hop integrity and authentication model. Fortunately, such usage mirrors the IP end-to-end model. In the case where RSVP is generating end-to-end messages and integrity and/or authentication are desired, the standard IPSEC based integrity and authentication methods MUST be used.

This draft introduces no other new security considerations to [RFC3209].

13. IANA Considerations

IANA assigns values to RSVP protocol parameters. Within the current document multiple objects are defined. Each of these objects contain C-Types. This section defines the rules for the assignment of the related C-Type values. This section uses the terminology of BCP 26 "Guidelines for Writing an IANA Considerations Section in RFCs" [BCP26].

As per [RFC2205], C-Type is an 8-bit number that identifies the function of an object. There are no range restrictions. All possible values except zero are available for assignment.

The assignment of C-Type values of the objects defined in this document fall into three categories. The first category inherit C-Types from the Label object, i.e., object class number 16 [RFC3209]. IANA is requested to institute a policy whereby all C-Type values assign for the Label object are also assigned for the following objects:

- o Suggested_Label (Class-Num TBA)
- o Upstream_Label (Class-Num TBA)
- o Recovery_Label (Class-Num TBA)

The second category of objects follow independent policies. Specifically, following the policies outlined in [BCP26], C-Type values in the range 0x00 - 0x3F are allocated through an IETF Consensus action, values in the range 00x40 - 0x5F are allocated as First Come First Served, and values in the range 0x60 - 0x7F are reserved for Private Use. This policy applies to the following objects.

- o Label_Set (Class-Num TBA)
- o Notify_Request (Class-Num TBA)
- o Protection (Class-Num TBA)
- o Admin Status (Class-Num TBA)
- o Restart_Cap (Class-Num TBA)

The assignment of C-Type values for the remaining object, the Acceptable_Label_Set object, follows the assignment of C-Type values of the Label_Set object. IANA is requested to institute a policy whereby all C-Type values assigned for the Label_Set object are also assigned for the Acceptable_Label_Set object.

13.1. IANA [Suggestions /] Assignments

[Note to RFC Editor:
Please drop all text enclosed in brackets in this section once IANA assignments are made.

Editor's Note:
The values in this section were suggested to IANA for assignment on March 11, 2002. No acknowledgment of the request has been received. The values are included for reference purposes only and should considered as unassigned.]

This section summarizes values used in this document that [will be /] have been assigned by IANA.

Message Types

o Notify message (suggested Message type =21)

Class Types

o RSVP_HOP (Existing C-Num 3)
  - IPv4 IF_ID RSVP_HOP (Suggested C-type =3)
  - IPv6 IF_ID RSVP_HOP (Suggested C-type =4)

o ERROR_SPEC (Existing C-Num 6)
  - IPv4 IF_ID ERROR_SPEC (Suggested C-type =3)
  - IPv6 IF_ID ERROR_SPEC (Suggested C-type =4)

o LABEL_REQUEST (Existing Class-Num 19)
  - Generalized_Label_Request (Suggested C-Type =4)

o RSVP_LABEL (Existing Class-Num 16)
  - Generalized_Label (Suggested C-Type =2)
  - Waveband_Switching_Label C-Type (Suggested C-Type =3)

New Class-Nums, C-Types inherited from Label object (same as CNum16)

o RECOVERY_LABEL Class-Num of form 0bbbbbbb (suggested =34)
o SUGGESTED_LABEL Class-Num of form 10bbbbbb (suggested =129)
o UPSTREAM_LABEL Class-Num of form 0bbbbbbb (suggested =35)

```
   ---------------------------------------------------------------------
   New Class-Nums

   o LABEL_SET                Class-Num of form 0bbbbbbb (suggested =36)
     - Type 1               (C-Type =1)
   o ACCEPTABLE_LABEL_SET     Class-Num of form 10bbbbbb (suggested =130)
     - Type 1 Acceptable_Label_Set (C-type from label_set cnum)
   o NOTIFY_REQUEST           Class-Num of form 11bbbbbb (suggested =195)
     - IPv4 Notify Request  (C-Type =1)
     - IPv6 Notify Request  (C-Type =2)
   o PROTECTION               Class-Num of form 0bbbbbbb (suggested =37)
     - Type 1               (C-Type =1)
   o ADMIN STATUS             Class-Num of form 11bbbbbb (suggested =196)
     - Type 1               (C-Type =1)
   o RESTART_CAP              Class-Num of form 10bbbbbb (suggested =131)
     - Type 1               (C-Type =1)

   ---------------------------------------------------------------------
   ERO/RRO subobject types

   o Label ERO subobject
     Type 3 - Label

   o Label RRO subobject
     Type 3 - Label

   ---------------------------------------------------------------------
   Error codes

   o "Routing problem/Label Set"                   (Suggested value =11)
   o "Routing problem/Switching Type"              (Suggested value =12)
                                          (duplicate code 13 dropped)
   o "Routing problem/Unsupported Encoding"        (Suggested value =14)
   o "Routing problem/Unsupported Link Protection" (Suggested value =15)
   o "Notify Error/Control Channel Active State"   (Suggested value =4)
   o "Notify Error/Control Channel Degraded State" (Suggested value =5)
   ---------------------------------------------------------------------
```

14. References

14.1. Normative References

[MPLS-UNNUM] Kompella, K., Rekhter, Y., "Signalling Unnumbered Links in RSVP-TE", Internet Draft, draft-ietf-mpls-rsvp-unnum-02.txt, August 2001

[GMPLS-SIG] Ashwood-Smith, P. et al, "Generalized MPLS - Signaling Functional Description", Internet Draft, draft-ietf-mpls-generalized-signaling-08.txt, April 2002.

[RFC2205] Braden, R. Ed. et al, "Resource ReserVation Protocol -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2961] Berger, L. et al, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001

[RFC3209] Awduche, et al, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

14.2. Informative References

[BCP26] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[MPLS-HIERARCHY] Kompella, K., and Rekhter, Y., "LSP Hierarchy with MPLS TE", Internet Draft, draft-ietf-mpls-lsp-hierarchy-02.txt, Feb., 2001.

[GMPLS-LDP] Ashwood-Smith, P. et al, "Generalized MPLS Signaling - CR-LDP Extensions", Internet Draft, draft-ietf-mpls-generalized-cr-ldp-06.txt, April 2002.

[PAN-RESTART] Pan, P, et al, "Graceful Restart Mechanism for RSVP-TE", Internet Draft, draft-pan-rsvp-te-restart-01.txt, July 2001.

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

15. Authors' Addresses

Peter Ashwood-Smith
Nortel Networks Corp.
P.O. Box 3511 Station C,
Ottawa, ON K1Y 4H7
Canada
Phone: +1 613 763 4534
Email: petera@nortelnetworks.com

Ayan Banerjee
Calient Networks
5853 Rue Ferrari
San Jose, CA 95138
Phone: +1 408 972-3645
Email: abanerjee@calient.net

Lou Berger Editor & Primary Point of Contact
Movaz Networks, Inc.
7926 Jones Branch Drive
Suite 615
McLean VA, 22102
Phone: +1 703 847-1801
Email: lberger@movaz.com

Greg Bernstein
Ciena Corporation
10480 Ridgeview Court
Cupertino, CA 94014
Phone: +1 408 366 4713
Email: greg@ciena.com

John Drake
Calient Networks
5853 Rue Ferrari
San Jose, CA 95138
Phone: +1 408 972 3720
Email: jdrake@calient.net

Yanhe Fan
Axiowave Networks, Inc.
200 Nickerson Road
Marlborough, MA 01752
Phone: + 1 774 348 4627
Email: yfan@axiowave.com

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

Jonathan P. Lang
Calient Networks
25 Castilian
Goleta, CA 93117
Email: jplang@calient.net

Fong Liaw
Solas Research, LLC
Email: fongliaw@yahoo.com

Eric Mannie
KPNQwest
Terhulpsesteenweg 6A
1560 Hoeilaart - Belgium
Phone: +32 2 658 56 52
Mobile: +32 496 58 56 52
Fax: +32 2 658 51 18
Email: eric.mannie@kpnqwest.com

Ping Pan
Juniper Networks
1194 N.Mathilda Ave
Sunnyvale, CA 94089
Email: pingpan@juniper.net

Bala Rajagopalan
Tellium, Inc.
2 Crescent Place
P.O. Box 901
Oceanport, NJ 07757-0901
Phone: +1 732 923 4237
Fax: +1 732 923 9804
Email: braja@tellium.com

Yakov Rekhter
Juniper Networks, Inc.
Email: yakov@juniper.net

Debanjan Saha
Tellium Optical Systems
2 Crescent Place
Oceanport, NJ 07757-0901
Phone: +1 732 923 4264
Fax: +1 732 923 9804
Email: dsaha@tellium.com

Vishal Sharma
Metanoia, Inc.
335 Elan Village Lane, Unit 203
San Jose, CA 95134-2539
Phone: +1 408-943-1794
Email: v.sharma@ieee.org

George Swallow
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA 01824
Voice: +1 978 244 8143
Email: swallow@cisco.com

Z. Bo Tang
Tellium, Inc.
2 Crescent Place
P.O. Box 901
Oceanport, NJ 07757-0901
Phone: +1 732 923 4231
Fax: +1 732 923 9804
Email: btang@tellium.com

Generated on: Thu Apr 25 12:57:53 2002