Network Working Group   Peter Ashwood-Smith (Nortel Networks Corp.)
Internet Draft   Ayan Banerjee (Calient Networks)
Expiration Date: October 2001   Lou Berger (Movaz Networks)
                                     Greg Bernstein (Ciena Corporation)
                                          John Drake (Calient Networks)
                                          Yanhe Fan (Axiowave Networks)
                                 Kireeti Kompella (Juniper Networks, Inc.)
                                                   Eric Mannie (EBONE)
                                   Jonathan P. Lang (Calient Networks)
                                      Bala Rajagopalan (Tellium, Inc.)
                                 Yakov Rekhter (Juniper Networks, Inc.)
                                        Debanjan Saha (Tellium, Inc.)
                                   Vishal Sharma (Jasmine Networks)
                                       George Swallow (Cisco Systems)
                                          Z. Bo Tang (Tellium, Inc.)

                                                        April 2001

         Generalized MPLS Signaling - RSVP-TE Extensions

                 draft-ietf-mpls-generalized-rsvp-te-02.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 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 CR-LDP specific description can be found in
   [GMPLS-LDP]. A generic functional description is presented in
   [GMPLS-SIG].

Contents

Changes from previous version:

o Fixed format of notification message. Had <upstream notify session>
   and <downstream notify session> reversed.
o Added Acceptable Label Set
o Removed SONET/SDH specifics.

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**

   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 format of a Generalized Label Request is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num (19)|C-Type (4)[TBA]|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | LSP Enc. Type |    Reserved   |             G-PID             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

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

The format of a Generalized Label is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | Class-Num (16)|  C-Type (2)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Label                             |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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. For compatibility reasons, a new RSVP c-type (3) is
   assigned for the Waveband Label.

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

```
    0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             Length            | Class-Num (16)|   C-Type (3)  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          Waveband Id                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          Start Label                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                           End Label                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   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. Without this
   mechanism and with an odd number of mirrored switching operations,
   the egress LSRs will not know that an input wavelength of say L1 will
   emerge from the waveband tunnel as L100.

   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 is identical to a generalized label.
It is used in Path messages. Suggested Label uses a new Class-Number
(TBA of form 10bbbbbb) and the C-type of the label being suggested.

Errors in received Suggested Labels MUST be ignored. This includes
any received inconsistent or unacceptable values.

2.5. Label Set

The Label_Set object uses a Class-Number TBA (of form 0bbbbbbb) and
the C-type of 1.

The format of a Label_Set is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num(TBA)|   C-Type (1)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Reserved           |  Label Type   |    Action     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Subchannel 1                          |
   |                            ...                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   :                              :                               :
   :                              :                               :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Subchannel N                          |
   |                            ...                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Label Type: 8 bits

   Indicates the type and format of the labels carried in the
   object. Values match the C-Type of the appropriate Label
   object.

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 has the same format as
   the generalized label, see Section 2.2. The Upstream Label uses
   Class-Number TBA (of form 0bbbbbbb) and the C-type of the label being
   suggested.

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 is added to the Path
   message. The Upstream Label 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 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/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
   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
   11bbbbbb). 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
```
       0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             Length            | Class-Num(TBA)| C-Type (1)    |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                  IPv4 Notify Node Address                     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

      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
```
       0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             Length            | Class-Num(TBA)|  C-Type (2)   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                              |
       |                  IPv6 Notify Node Address                    |
       |                                                              |
       |                                                              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

      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 only generated 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
[RSVP-RR] is used to acknowledge the receipt of a Notify Message.
See [RSVP-RR] 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 Msg 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> [<POLICY_DATA>...]
                                  <sender descriptor>

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

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 [RSVP-RR] and are used when refresh
reductions is supported.

4.3.2. Procedures

Notify messages are 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 [RSVP-RR].

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 subobject is defined as follows:

        0   1   2   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+
       |L| Type | Length |U| Reserved | C-Type |
       +-+
       | Label |
       | ... |
       +-+

   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. **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. The preceding subobject must be a strict
   object. 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 it's 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.

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 a Class-Number TBA (of form 0bbbbbbb).

The format of Protection Information Object is:

```
   0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Length             | Class-Num(TBA)|  C-Type (1)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |S|                  Reserved                      | Link Flags|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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. 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
[RSVP-RR].

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> ]
                         [ <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> ]
```

The format of the sender description for bidirectional LSPs is:

```
<sender descriptor> ::=  <SENDER_TEMPLATE> <SENDER_TSPEC>
                         [ <ADSPEC> ]
                         [ <RECORD_ROUTE> ]
                         [ <SUGGESTED_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> ]
                          [ <POLICY_DATA> ... ]
                          <STYLE> <flow descriptor list>
```

     <flow descriptor list> is not modified by this document.

The format of a Resv 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>
```

8. 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

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

9. Security Considerations

   The transmission of notify messages using IP in IP, break 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 SHOULD be used.

   This draft introduces no other new security considerations to [RSVP-
   TE].

10. References

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

[MPLS-UNNUM] Kompella, K., Rekhter, Y., "Signalling Unnumbered Links
              in RSVP-TE", Internet Draft,
              draft-ietf-mpls-rsvp-unnum-01.txt, February 2001

[GMPLS-LDP] Ashwood-Smith, P. et al, "Generalized MPLS Signaling -
            CR-LDP Extensions", Internet Draft,
            draft-ietf-mpls-generalized-cr-ldp-01.txt,
            February 2001.

[GMPLS-SIG] Ashwood-Smith, P. et al, "Generalized MPLS -
            Signaling Functional Description", Internet Draft,
            draft-ietf-mpls-generalized-signaling-02.txt,
            February 2001.

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

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

[RSVP-TE] Awduche, D.O., Berger, L., Gan, D.-H., Li, T., Swallow, G.,
          and Srinivasan, V., "RSVP-TE: Extensions to RSVP for LSP
          Tunnels," Internet Draft,
          draft-ietf-mpls-rsvp-lsp-tunnel-08.txt, February 2001.

[RSVP-RR]   Berger L., Gan D., Swallow G., Pan P., Tommasi F.,
            Molendini S., "RSVP Refresh Overhead Reduction Extensions",
            draft-ietf-rsvp-refresh-reduct-05.txt, June 2000.

11. 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
    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.
100 Nickerson Road
Marlborough, MA 01752
Phone: +1 508 460 6969 Ext. 627
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

Eric Mannie
EBONE
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@ebone.com

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
Jasmine Networks, Inc.
3061 Zanker Road, Suite B
San Jose, CA 95134
Phone: +1 408 895 5030
Fax: +1 408 895 5050
Email: vsharma@jasminenetworks.com

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: Mon Apr 16 19:44:44 EDT 2001