INTERNET DRAFT
draft-yasukawa-mpls-rsvp-multicast-01.txt
Expires: May 2003

Seisho Yasukawa
Masanori Uga
Hisashi Kojima
Koji Sugisono
NTT
Alan Kullberg
Netplane Systems
Markus Jork
Avici Systems
Dimitri Papadimitriou
Alcatel

November 2002

Extended RSVP-TE for Multicast LSP Tunnels
<draft-yasukawa-mpls-rsvp-multicast-01.txt>

0. Summary for Sub-IP Area

(This section to be removed before publication.)

0.1. Summary

This document specifies extensions and mechanisms to RSVP-TE in support of MPLS point to multipoint LSPs.

0.2. Where does it fit in the Picture of the Sub-IP Work

This work fits squarely in the MPLS box.

0.3. Why is it Targeted at this WG

This draft is targeted at the MPLS WG, because this draft specifies the extensions to RSVP-TE signalling protocol in support of MPLS point to multipoint LSPs creation/deletion and Leaf initiated Join/Leave signalling.

0.4. Justification

In this draft the protocol extensions will be defined allowing the creation and deletion of multicast trees by other means than pure multicast routing protocols. The reasons for enabling this are:

1. Some network operators don't inject BGP routes into their backbone (e.g. they create a full mesh of LSPs). The result is that SSM can not be deployed, because the intermediate nodes in the network do not know where to send the PIM-SM Join messages to (if the source is in another AS).

2. Some applications of multicast do not require very dynamic trees, e.g. content distribution to proxy servers or subtrees in backbone networks. For these applications it becomes worthwhile to create more optimal distribution trees. Multicast trees constructed by multicast routing protocols are not optimal because:

a. These trees are only shortest path if the paths are symmetric. This is a false assumption in the current Internet.

b. Shortest-path trees themselves are non-optimal. On the other hand the calculation of an optimal Steiner trees is known to be an NP-complete problem requiring the usage of heuristics.

3. In a next step - similar to unicast traffic engineering [RFC-2702] - an MPLS multicast tree (a point-to-multipoint LSP) can be built which is automatically computed by a suitable entity based on QoS and policy requirements, taking into consideration the network state. In this case an IGP is needed. Both, OSPF [OSPF-TE] and IS-IS [ISIS-TE] can be used for this purpose. The LSA information is flooded throughout an AS, the point-to-multipoint tree is then calculated based on the adequate topology.

0.5. Related I-d's

- draft-poj-optical-multicast-02.txt (expired)

  This i-d addresses the specifics of optical point-to-multipoint connection, while focusing on non-packet environments only this i-d is the early precursor of the Tree ERO object and the related mechanisms developed in the present i-d.

- draft-ooms-mpls-multicast-te-01.txt (expired)

  This i-d describes 2 ways of building a multicast traffic-engineered tree: root-initiated tree and leaf-initiated tree. It also proposed defines extensions to MPLS (CR-LDP) signalling for setting up and tearing down traffic engineered multicast trees.

- draft-cheng-mpls-rsvp-multicast-er-00.txt (expired)

  This i-d introduces the RSVP-TE's explicit route capability in support of multicast applications and more generally to point-to multipoint LSPs. The proposed mechanisms were also intended for point-to multipoint applications in non-packet LSP capable networks. First i-d having clarified the processing of the Tree ERO object and initiate corresponding RSVP-TE message processing.

- draft-chung-mpls-rsvp-multicasting-00.txt (expired)

  This i-d addresses extensions to the Resource Reservation Protocol (RSVP) to support MPLS multicast. The concepts developed are in accordance to the traffic engineering (TE) attributes provided in the MPLS-TE specifications. This i-d introduces the concept of Leaf initiated Join/Leave procedures using RSVP-TE.

Table of Contents

1. Introduction

Multicast technology will become increasingly important with the dissemination of new applications such as contents delivery services and video conferences, which require much more bandwidth and stricter QoS than conventional applications. From the service providers' perspective, traffic engineering (TE) functions will be needed to handle the large amount of multicast traffic.

This document defines some protocol extensions to the existing RSVP-TE[1] in order to establish a multicast label switched path (LSP). The use of label switching routers (LSRs) with the protocol extensions defined in this document allows service providers to offer unicast and multicast multiprotocol label switching (MPLS) services in the same service network.

This protocol assumes a variable LSP topology, e.g., point-to-multipoint topologies. This document describes how to establish and maintain point-to-multipoint LSPs as the most basic multicast topology. It defines two ways of constructing a point-to-multipoint LSP: sender-initiated LSP setup and leaf-initiated LSP setup. Each method has an LSP modification capability in order to adapt to dynamic changes in the LSP tree topology.

The establishment of multipoint-to-multipoint LSPs is a subject for future study.

This protocol is very flexible and can be used to carry protocols other than IP multicasting, e.g., Ethernet, PPP, and SONET/SDH. No assumption is made about the format of the data to be carried in the signaled LSP.

2. Terminology

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 [5].

The reader is assumed to be familiar with the terminology in [1],

[2], [3] and [4].

Multicast LSP:

A label switched path that has more than one egress label switching router

sender node:
Headend of the multicast LSP. It controls the multicast LSP layout and places traffic onto it by pushing a label.

Branch LSR:

A label switching router (LSR) that has more than one downstream LSR. A branch LSR receives a single MPLS frame, makes a duplicate of it, and sends each to downstream interfaces.

join leaf node:

A leaf node trying to join a multicast LSP.

leave leaf node:

A leaf node trying to leave a multicast LSP that it has joined

graft node:

An LSR that is already a member of the multicast tree and is in process of signaling a new subtree.

prune node:

An LSR that is already a member of the multicast tree and is in process of tearing down an existing subtree.

Explicit Route Object (ERO):

An object specifying the explicit path of the message including the object.

Tree Explicit Route Object (TERO):

An extended ERO for describing the multicast tree topology.

Record Route Object (RRO):

An object recording the information of route through which the message including the object has passed.

Tree Record Route Object (TRRO):

An extended RRO for recording the route along which each merged message has traveled.

Subtree:

A subtree is a portion of a multicast tree starting at a particular node that is a member of the multicast tree and includes ALL nodes downstream of that node that are also members of the multicast tree.

3. Architecture

3.1 Multicast LSP tunnels

This protocol defines "multicast flow" by a label that is assigned to a set of packets at the ingress node of a point-to-multipoint LSP. Such a point-to-multipoint LSP is referred to as a "multicast LSP tunnel" because the traffic through it is opaque to intermediate nodes along the LSP. To enable the identification and association of such multicast LSP tunnels, new MULTICAST_LSP_TUNNEL session objects are defined. Each session object carries the sender node address of a point-to-multipoint LSP and its tunnel ID. The sender node address is more preferable than destination (leaf) node addresses to identify a multicast LSP tunnel because frequent topological changes may occur and destination node addresses are not stable in this tunnel.

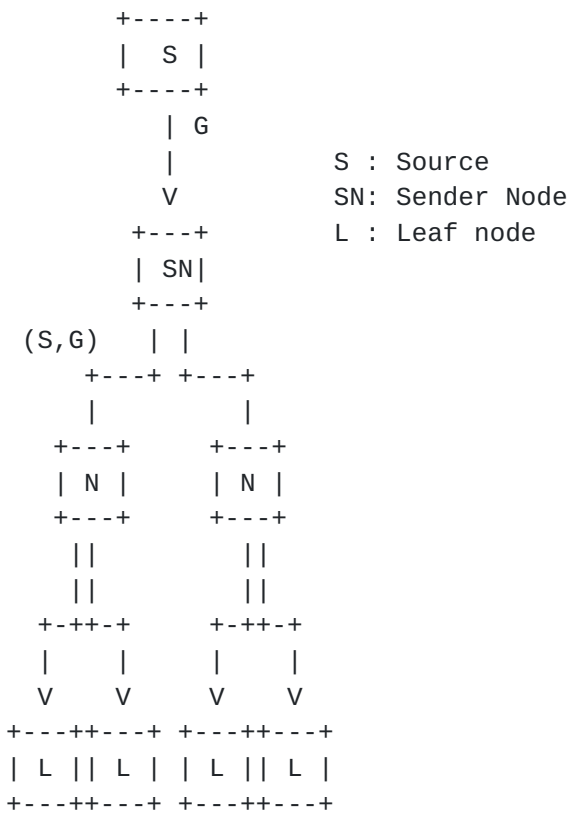
And to express the leaf nodes and topology of this multicast LSP tunnel, TREE_EXPLICIT_ROUTE object and TREE_RECORD_ROUTE object are also defined. This protocol uses conventional SENDER_TEMPLATE (or FILTER_SPEC) object to convey sender node address and LSP ID. Therefore, the MULTICAST_LSP_TUNNEL session object together with the SENDER_TEMPLATE object uniquely identify a multicast LSP tunnel.

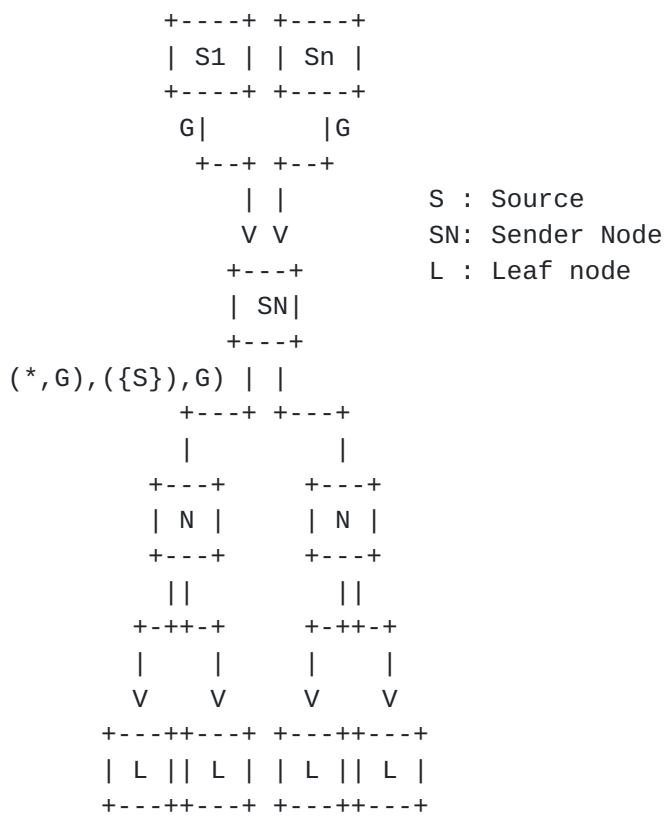
It is very useful to associate sets of LSP tunnels which share the same sender node in a multicast application. This can be useful during rerouting operations or to spread a traffic trunk over multiple multicast paths [1]. In traffic engineering such sets are called multicast traffic engineered tunnels (multicast TE tunnels). As same as unicast TE tunnels, these multicast TE tunnels is uniquely identified by the SESSION objects.

3.2 Multicast LSP topology

The proposed RSVP-TE extensions support point-to-multipoint LSP tunnel establishment, teardown, and modification mechanisms. A point-to-multipoint LSP can handle i) (S,G) multicast traffic, ii)

({S},G), (*,G) multicast traffic, and iii) ({S},{G}) multicast traffic. (S,G) represents a single sender node with source address S transmitting multicast traffic to a multicast group G. The ({S},G) represents several sender nodes with source addresses {S} transmitting multicast traffic to a multicast group G. The (*,G) represents arbitrary sender nodes transmitting multicast traffic to a multicast group G. The ({S},{G}) represents several sender nodes with source addresses {S} transmitting multicast traffic to multicast groups {G}. Traffic that shares the same multicast distribution topology and is treated in the same forwarding manner is defined as belonging to the same FEC.




S1
Sn
G
G
S : Source
SN: Sender Node
L : Leaf node
SN
(*,G),({S}),G)
N
N
L
L
L
L

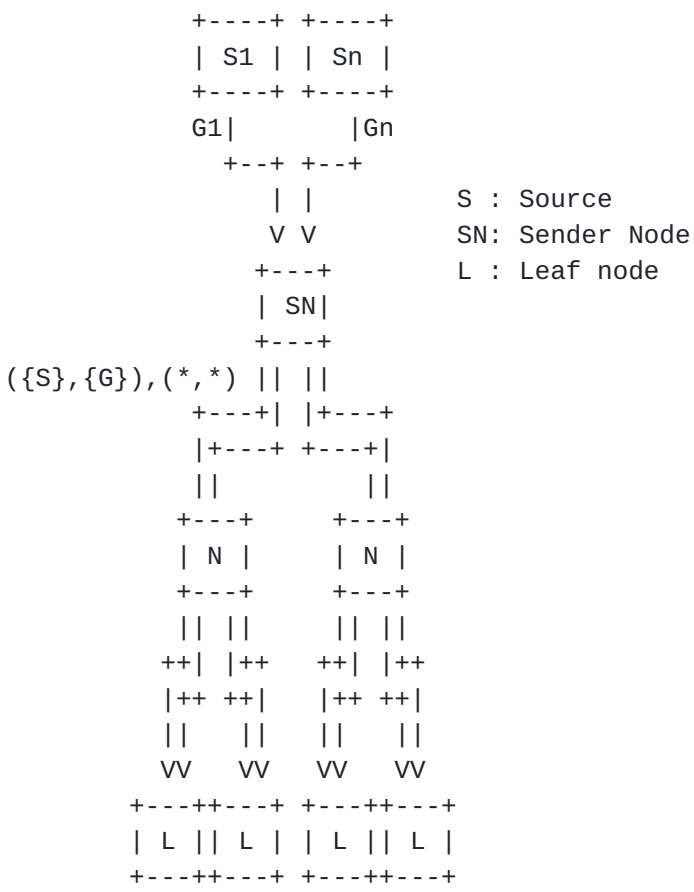


Figure 1: Multicast traffic type

3.3 Calculation of multicast tree route

There are two methods of calculating the multicast tree route. It can be calculated by: i) the sender node or the network management system (NMS) and ii) a IP multicast routing protocol such as PIM-SM[8]. The first method is suitable to achieve traffic engineering because we can control multicast LSP topology independent to existing IP routing protocol. This means that we can setup multicast LSP and control its topology within a unicast IP network without using a complicated IP multicast routing protocol. Multicast CSPF [13] calculation at a sender node using a conventional unicast routing with TE extension [14] is one of the example.

On the other hand, the second method is suitable to easy implementation. But it is necessary to establish a method for cooperating with multicast routing protocol. And it is not easy to control multicast LSP topology because conventional IP multicast routing protocol such as PIM-SM frequently changes its multicast route. And it is difficult to convey a several multicast traffic that has different multicast group addresses in a single multicast LSP

because it is setup per group address basis.

This draft assume the first method for multicast tree calculation and the second method is a topic for further study.

3.4 Multicast LSP establishment, teardown, and modification mechanisms

This multicast MPLS protocol can be applied to various multicast applications because it supports both sender-initiated and leaf-initiated multicast LSP tunnel establishment mechanisms. It is preferable for multicast sender nodes to start traffic distribution only after they have confirmed the establishment of an end-to-end LSP. For this, we can use the sender-initiated multicast LSP tunnel establishment mechanism. This mechanism is also preferable when establishing a QoS-capable multicast LSP tunnel because the single sender node can calculate the optimum multicast LSP tunnel using QoS information.

On the other hand, some applications need a leaf-initiated multicast LSP tunnel establishment mechanism. A leaf node that implements IGMP[6,7] protocol in addition to this multicast MPLS protocol can dynamically invoke the multicast LSP tunnel setup mechanism according to the multicast client's content viewing status. If all the clients accommodated at the leaf node stop receiving multicast traffic from this node (e.g., all clients send a IGMP leave message to the leaf node), the leaf node invokes this process and requests to tear down unnecessary multicast LSP tunnels. If a client accommodated at the leaf node starts receiving multicast traffic from this node (e.g., a client who wants to receive this multicast data sends an IGMP Join message to the leaf node), the leaf node invokes this process to graft the necessary multicast LSP tunnel.

We must assume the co-existence of both applications in this multicast MPLS network. Therefore, the multicast MPLS protocol supports both sender- and leaf-initiated multicast LSP tunnel establishment mechanisms, and these mechanisms must work in an interoperable manner.

Traffic engineering is more important in a multicast network environment than a unicast one. Frequent topological changes may occur in a multicast LSP tunnel because it has many leaf nodes and a more complex transmission topology. We need a more reliable tunnel re-establishment mechanism in a multicast network environment because the nodes nearer the sender node need greater reliability than ones placed at leaf neighbors. For these demands, a partial multicast LSP tunnel modification mechanism (subtree expansion and reduction mechanism) is necessary because total multicast LSP tunnel re-establishment is not efficient in a multicast network. Therefore,

the proposed RSVP-TE extensions implement a partial multicast LSP tunnel modification mechanism.

The features supported by the proposed RSVP-TE extensions are summarized below. The sender-initiated multicast LSP tunnel establishment mechanism has i) multicast LSP tunnel establishment and teardown mechanisms and ii) partial multicast LSP (subtree LSP) tunnel establishment, teardown, and modification mechanisms. The leaf-initiated multicast LSP tunnel establishment mechanisms has iii)leaf LSP tunnel establishment, teardown mechanism.

The proposed RSVP-TE extensions use an interoperable architecture between the sender- and leaf-initiated multicast LSP tunnel establishment mechanisms, so both mechanisms can modify a multicast LSP tunnel established by either mechanism.

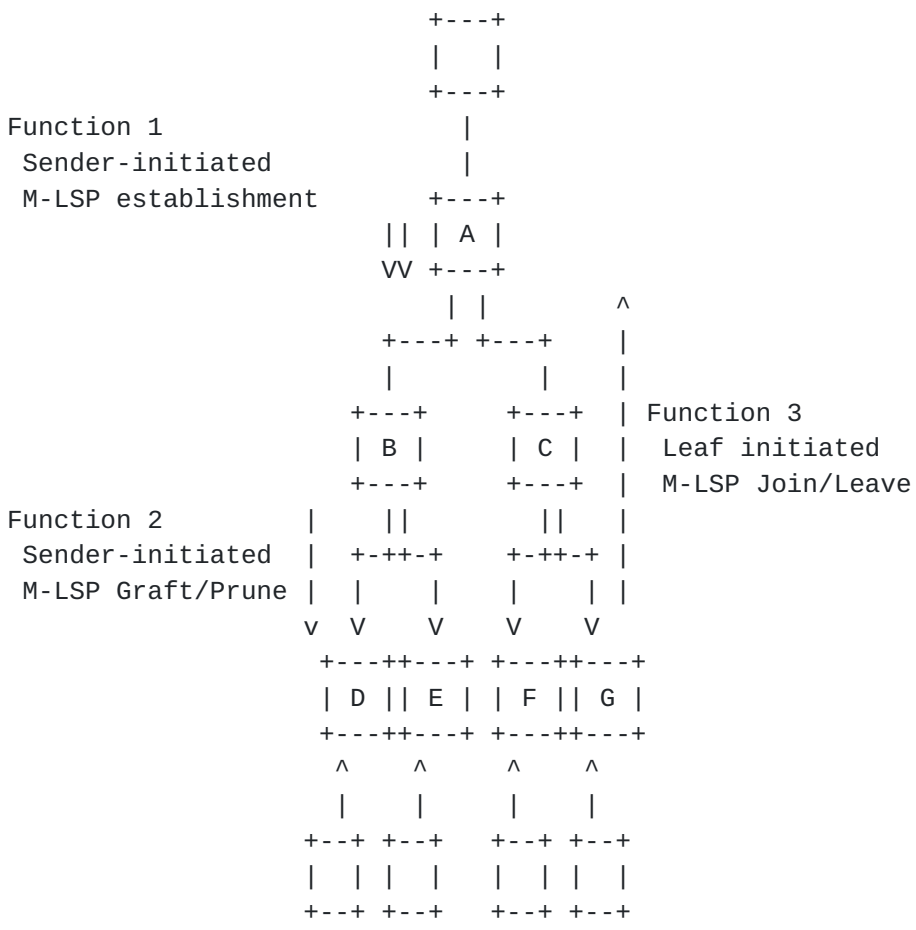


Figure 2: Function supported by this protocol

3.5 Basic operation of multicast LSP tunnels

This paragraph explains the basic multicast LSP tunnel establishment

mechanism. Figure 3 shows the basic sender-initiated multicast LSP tunnel establishment mechanism.

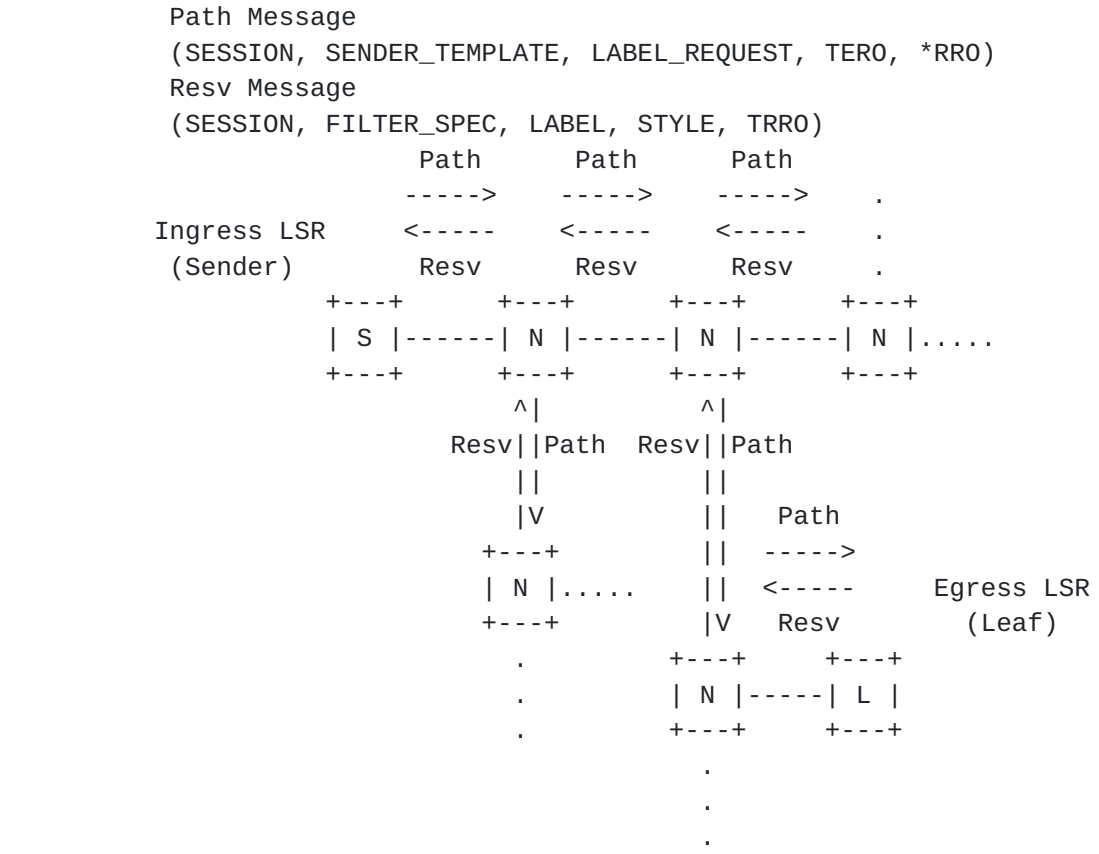


Figure 3: Fundamental Multicast LSP establish mechanism

To create a multicast LSP tunnel, the sender node creates a Path message with a TERO in addition to a LABEL_REQUEST object, a SESSION object, and a SENDER_TEMPLATE object. The TERO includes multicast tree information to set. The Path message is forwarded towards its destination along a multicast path specified by the TERO. Each intermediate node along the path records the TERO, SESSION object, and SENDER_TEMPLATE object in its path state block. An intermediate branch node copies the Path message to each next hop node specified in the TERO until the destination leaf nodes are reached.

When the Path message reaches the destination leaf node, the leaf node responds to a LABEL_REQUEST by including a LABEL object in its response Resv message. Then the Resv message is sent back upstream toward the sender node, following the path state created by the Path message in reverse order.

Each node that receives a Resv message containing a LABEL object uses that label for outgoing traffic associated with this tunnel. If the node is a branch node, it waits for all the Resv massages coming from

downstream leaf nodes. After receiving all the Resv messages, it allocates a new label and places it in the corresponding LABEL object of the Resv message, which it sends upstream to the previous hop (PHOP). Finally, the merged Resv message propagates upstream to the sender node. Thus, a multicast LSP is established. This label assignment is performed in downstream on-demand ordered control mode.

3.6 Multicast session

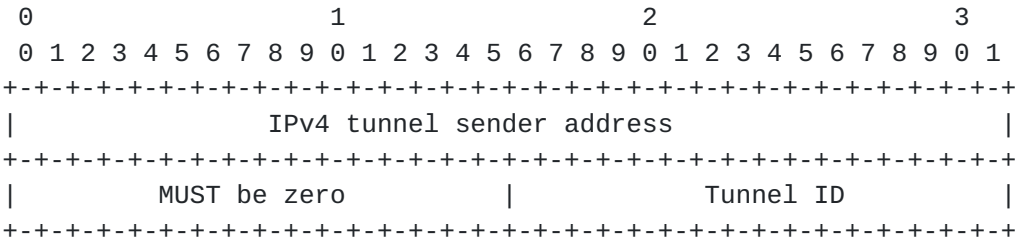
3.6.1 Multicast session object

The new SESSION object is defined for multicasting.

To identify a multicast tunnel, MULTICAST_LSP_TUNNEL_IPv4 and MULTICAST_LSP_TUNNEL_IPv6 are added to the SESSION object as new C-Types. They each have a tunnel sender address and Tunnel ID.

3.6.2 MULTICAST_ LSP_TUNNEL_IPv4 session objects

Class = SESSION, C-Type = MULTICAST_ LSP_TUNNEL_IPv4



IPv4 tunnel sender address

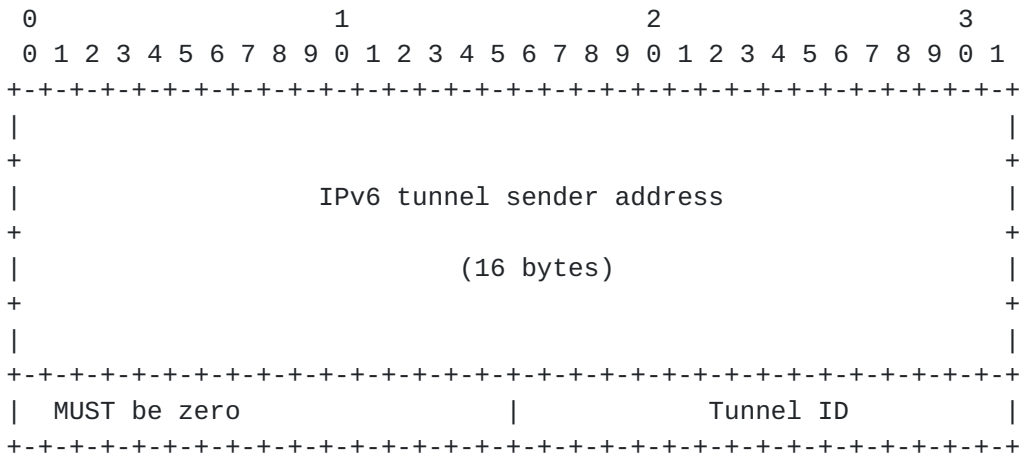
IPv4 address of the tunnel sender

Tunnel ID

A 16-bit identifier used in the SESSION that remains constant over the life of the tunnel.

3.6.3 MULTICAST_LSP_TUNNEL_IPv6 session objects

Class = SESSION, C-Type = MULTICAST_LSP_TUNNEL_ IPv6



IPv6 tunnel sender address

IPv6 address of the tunnel sender

Tunnel ID

A 16-bit identifier used in the SESSION that remains constant over the life of the tunnel.

3.7 Explicit routing

3.7.1 Tree Explicit Route Object (TERO)

3.7.1.1 Overview

Explicit routing is the main function of RSVP-TE. RSVP-TE defines an ERO to describe the explicit route of the tunnel. An ERO is defined as a list of subobjects. Each subobject represents information about nodes composing the data path. For multicasting, the path topology is a tree. So we extend ERO to express a multicast tree and define a new Tree Explicit Route Object (TERO).

TERO is included in a message concerning path establishment like the Path message. It specifies nodes of a tree in which the message is traversed. In particular, TERO of a Path message initiated by the sender node contains the whole topology of the multicast tree.

Before a node sends a TERO in an outgoing PATH message, it MUST split the TERO such that only hops describing the subtree towards which the PATH is sent are included in the TERO. The result is that downstream

nodes do not have a complete picture of the multicast tree. Instead, each node receives a TERO containing a subtree with the receiving node as the root of that subtree. By not sending the entire TERO downstream, downstream nodes will use less memory for storing the state information for the LSP.

TERO is also a list of subobjects. Each subobject shows information about a node on the multicast tree. To describe the tree topology, each subobject should be sorted to be able to show link connectivity. For this purpose, the subobjects are arranged in depth-first-order and have a number of hops from the sender node. For the multicast tree shown in figure 4 below, the TERO is encoded as follows:

T={A(0),B(1),C(2),D(3),E(3),F(2),G(2),H(3),I(1),J(1),K(2),L(2)}
Distance in parentheses.

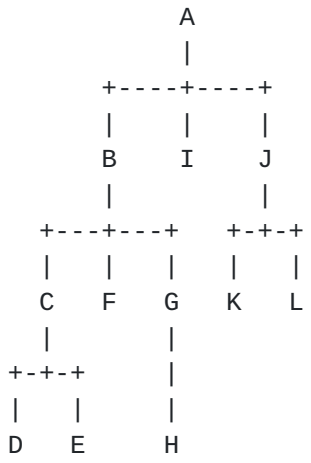


Figure 4: TERO corresponding to a multicast tree

3.7.1.2 Data Terminating Nodes

In some cases, a multicast tree may be calculated that contains a branch node that must also behave as a leaf node with respect to the dataplane (i.e., in the case of locally attached client nodes). Allowing for a node to be both a branch and leaf at the same time may enable calculation of otherwise impossible and/or more optimal trees.

In order for a node in the tree to know to terminate the data locally as well as forward the data to other downstream nodes, an explicit designator is placed in the TERO hop object. When set, a node SHOULD terminate data locally and also send it downstream.

3.7.1.3 Strict and loose explicit routes

Two kinds of explicit route are prepared. In a strict explicit route, the nodes composing the route are specified strictly. A loose explicit route allows an external routing algorithm to decide the route between specified nodes. Unicast routing algorithm such as OSPF[15] and IS-IS[16] are examples of such algorithms. These routes are selected at each link of the tree. The protocol allows a mixture of strict and loose explicit routes in the same multicast tree, so TERO should show where the strict and loose explicit routes are. The information should relate to nodes described by subobjects. As the information satisfying this condition, the input link status is used to specify strict and loose explicit routes in point-to-multipoint trees. In the case of a multicast point-to-multipoint tunnel, the attribute of an input link relates to each node uniquely. So as the information satisfying this condition, the input link status is used to specify strict and loose explicit routes in a point-to-multipoint tunnel. The L bit in a subobject of TERO shows the input link status of the node. If the L bit shows loose, the input link belongs to a loose explicit route. Otherwise, it belongs to a strict explicit route.

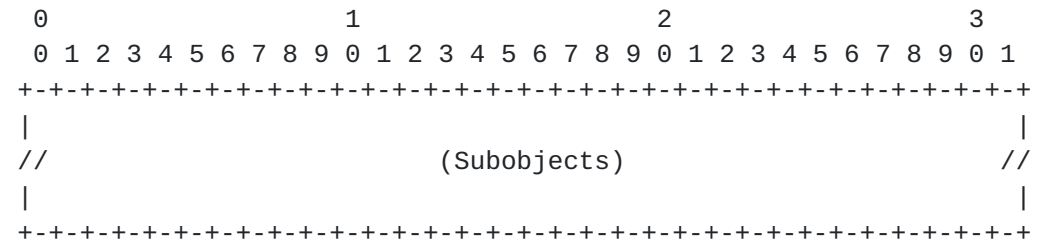
This protocol allows the insertion of additional subobjects. In a loose explicit route, the edge nodes of the route indicated as a loose explicit route may know the topology of the network around the loose explicit route. The edge node may calculate the route and specify the route with TEROs. In this case, the edge node inserts the calculated route into the TERO of messages related to the loose explicit route.

Consider the number of hops from the sender node to a node. It is counted based on the TERO. If intermediate nodes between a leaf node and the sender node belong to a strict explicit route, they are specified in TERO and the leaf node can count the number of hops. When loose explicit routes are part of a multicast tree, the sender node cannot count the number of hops. Because nodes on a loose explicit route are not specified, sender nodes cannot count the number of hops in the loose explicit route. So they cannot calculate the relevant number of hops of the tree nodes. Therefore, the protocol defines the number of hops between the edges of an loose explicit route as 1. This definition satisfies the demand for TERO; using this definition, TERO can show the tree topology and node connectivity.

The distance field in TERO hops is always relative to the sender node. This is true in ALL messages in which a TERO can appear.

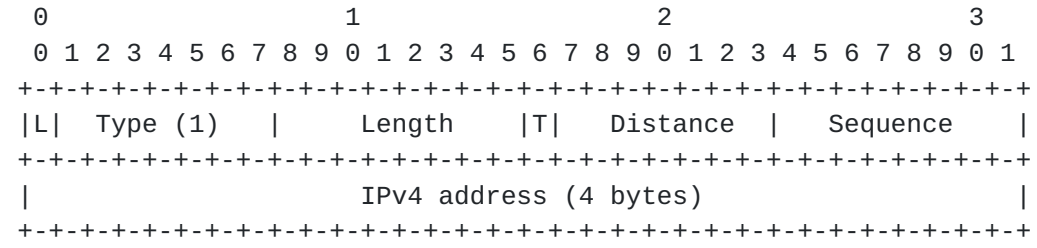
3.7.1.4 Object format

Class = TREE_EXPLICIT_ROUTE, C_Type = 1

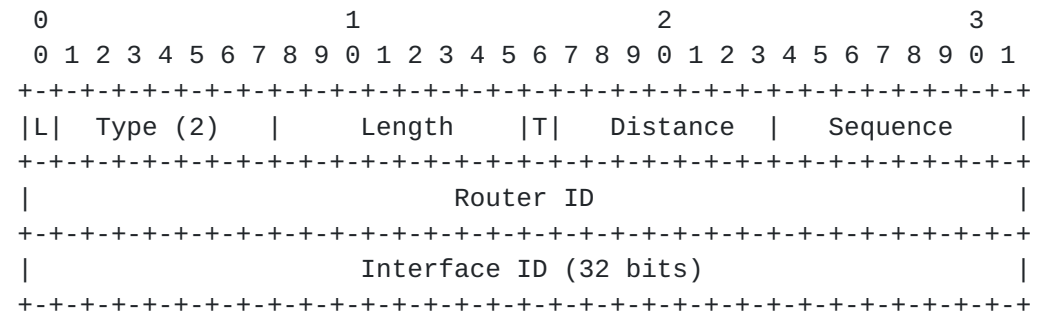


Subobject format

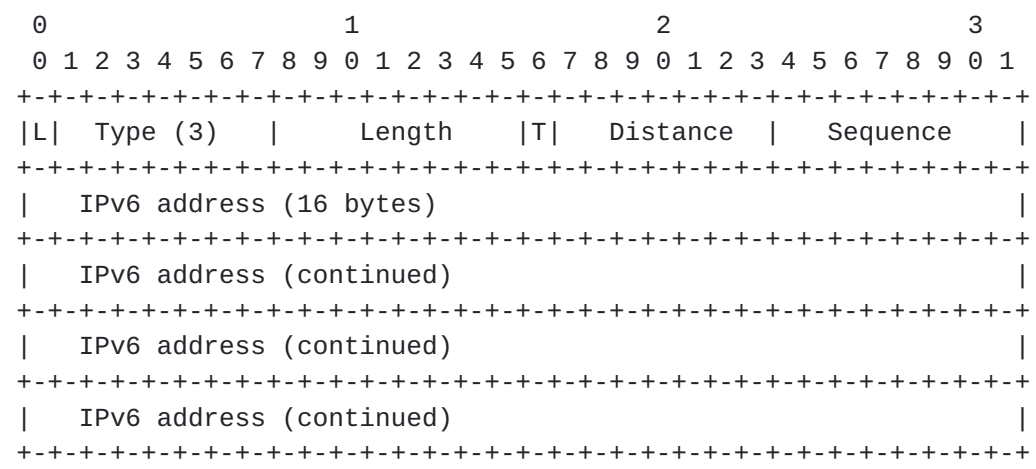
Type = IPv4_ADDRESS



Type = UNNUMBERED_INTERFACE



Type = IPv6_ADDRESS



L

The L bit is an attribute of the subobject. The L bit is set if the route of a path between the node specified by this subobject and its upstream node is not specified (Loose explicit route). If the bit is not set, the route of a path is specified explicitly (Strict explicit route).

Type

0x01 IPv4 address
0x02 Unnumbered Interface ID
0x03 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

IPv4/6 address

An IPv4/6 address of node corresponding to the subobject. In the case that the node has several IP addresses, the address of input interface should be specified.

Router ID

The Router ID of the node corresponding to the subobject.

Unnumbered Interface ID

The unnumbered interface ID [11].

Distance

The distance from the sender node to the node specified by the subobject. The value of distance between neighboring nodes specified in TERO is 1.

T

The T bit (terminating node) is set to 1 to indicate that the node in this hop is a data terminating node and has locally attached clients (receivers of the data). This bit MUST be set for any of the leaf nodes listed in the TERO.

Sequence

The sequence number associated with this TERO hop.

3.7.2 Tree Record Route Object (TRRO)

Record Route Object (RRO) is a list of subobjects describing a node. Each subobject is sorted to show the order in which the message went through the nodes. In multicast communication, some messages SHOULD be merged into a new message to integrate the information that each message conveys. In this case, RRO shows all the nodes through which each message traveled. So RRO should be able to represent the tree structure in multicasting. We call the extended RRO a TRRO.

When messages are merged at a branch node, their TRROs also are merged. Each TRRO shows the topology information of the downstream subtree. The merged TRRO is a series of downstream TRROs. The subobject specifying the branch node is inserted at the top of the series. So the merged TRRO shows the subtree whose root is the branch node. In the case of a Resv message, when the message arrives at the sender node of the multicast tree, TRRO shows the current topology of the multicast tree. To represent the tree topology, the subobjects of TRRO are sorted in depth-first-order and they have information about the number of hops from the sender node as the case of TERO. The topology information may be sent to all nodes composing the multicast tree by refreshing the Path message.

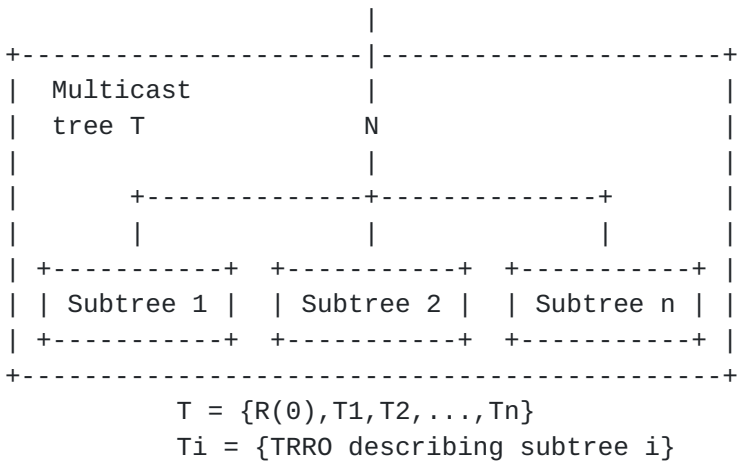
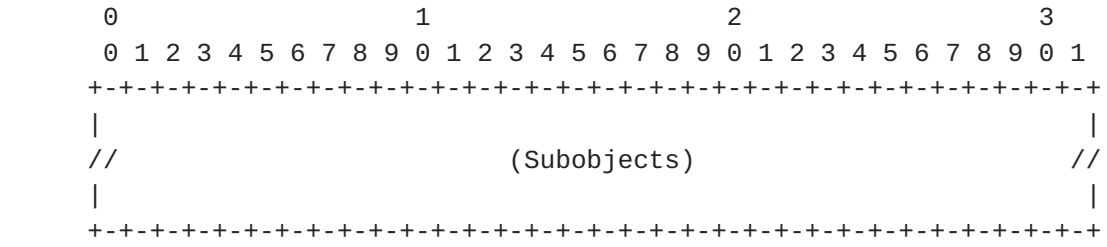


Figure 5 Merged TRRO

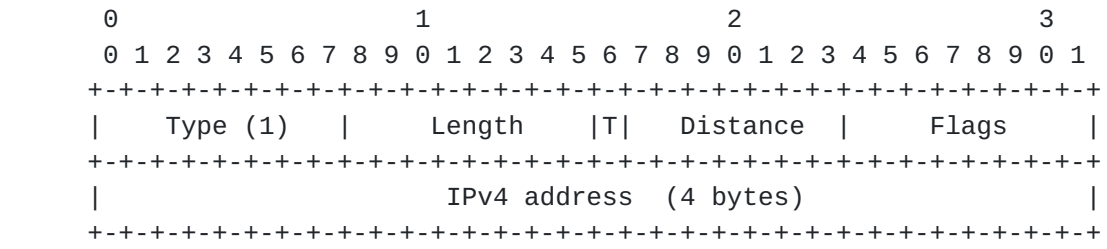
3.7.2.1 Object format

Class = TREE_RECORD_ROUTE, C_Type = 1

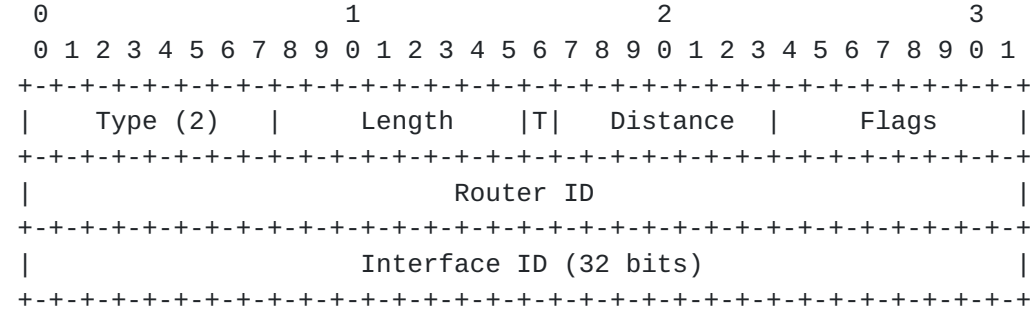


3.7.2.2 Subobject format

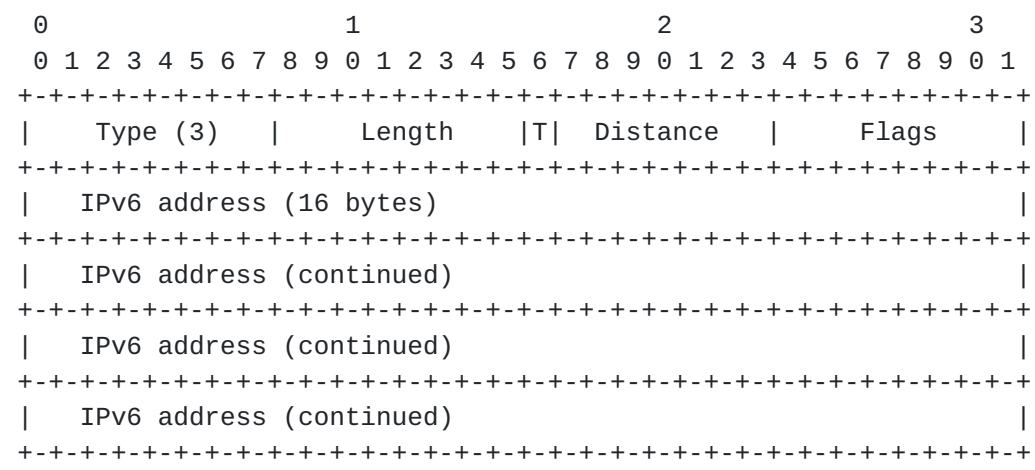
Type = IPv4_ADDRESS



Type = UNNUMBERED_INTERFACE



Type = IPv6_ADDRESS



Type

0x01 IPv4 address
0x02 Unnumbered Interface ID
0x03 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

T

The T bit (terminating node) is set to 1 to indicate that the node in this hop is a data terminating node and has locally attached clients (receivers of the data). This bit MUST be set for any of the leaf nodes listed in the TRRO.

Distance

The distance from sender node to the node specified by the subobject. The value of distance between neighboring nodes specified in TRRO is 1.

Flags

0x01 Local protection available

Indicates that the link downstream of this node is protected via a local repair mechanism. This flag can only be set if the Local protection flag was set in the SESSION_ATTRIBUTE object of the corresponding Path message.

0x02 Local protection in use

Indicates that a local repair mechanism is in use to maintain this tunnel (usually in the face of an outage of the link it was previously routed over).

0x04 Bandwidth protection

The PLR will set this when the protected LSP has a backup path which provides the desired bandwidth, which is that in the FAST_REROUTE object or the bandwidth of the protected LSP, if no FAST_REROUTE object was included. The PLR may set this whenever the desired bandwidth is guaranteed; the PLR MUST set this flag when the desired bandwidth is guaranteed and the "bandwidth protection desired" flag was set in the SESSION_ATTRIBUTE object.

0x08 Node protection

When set, this indicates that the PLR has a backup path providing protection against link and node failure on the corresponding path section. In case the PLR could only setup a link-protection backup path, the "Local protection available" bit will be set but the "Node protection" bit will be cleared.

IPv4/6 address

A 32/128-bit unicast, host address.

Router ID

The Router ID of the node corresponding to the subobject.

Unnumbered Interface ID

The unnumbered interface ID [11].

3.7.3 Message Size

The introduction of the TERO and TRRO objects has increased the likelihood that a message carrying either of these objects may exceed the link MTU. If a message exceeds the link MTU, then IP fragmentation and reassembly MUST be used in sending the message to it's destination.

3.8 Multicast Notify Message

The Multicast Notify message is used to convey information to non-adjacent nodes about multicast tree operations. Other sections of this document specify when it is appropriate to send the Multicast Notify message.

The contents of the Multicast Notify message is as follows:

```
<MulticastNotify Message>::=<Common Header>
                            <SESSION>
                            <SENDER_TEMPLATE>
                            <MULTICAST_NOTIFY>
```

The MULTICAST_NOTIFY object has 3 different C-Types as follows:

1 - MulticastErr
2 - JoinErr
3 - LeaveErr

The details of the contents of the MULTICAST_NOTIFY object for each of these C-Types are described in other sections of this document.

4. Sender-initiated multicast LSP establishment

4.1 Sender-initiated Multicast LSP establishment mechanism

The sender node initiates a Path message with a TERO including a multicast tree topology to be established. This Path message is forwarded along the path described in the TERO and duplicated by each branch node. Each node receiving the Path message records the multicast tree state described in the SESSION object and the SENDER_TEMPLATE object. A leaf node which receives a Path message responds to it by sending a Resv message including a LABEL and a TRRO object. A Resv message is sent back upstream towards the sender node on a hop-by-hop basis according to the recorded multicast tree state

in each intermediate node.

The Resv messages from all downstream nodes MUST be merged at the branch node. When the sender node receives the Resv messages, the establishment of the multicast LSP is completed. Figure 6 shows the sequence of multicast LSP establishment events. Node B, which is the branch node for the nodes C, D, and E, merges the Resv messages. After this merging, the node B initiates a Resv message and sends it upstream.

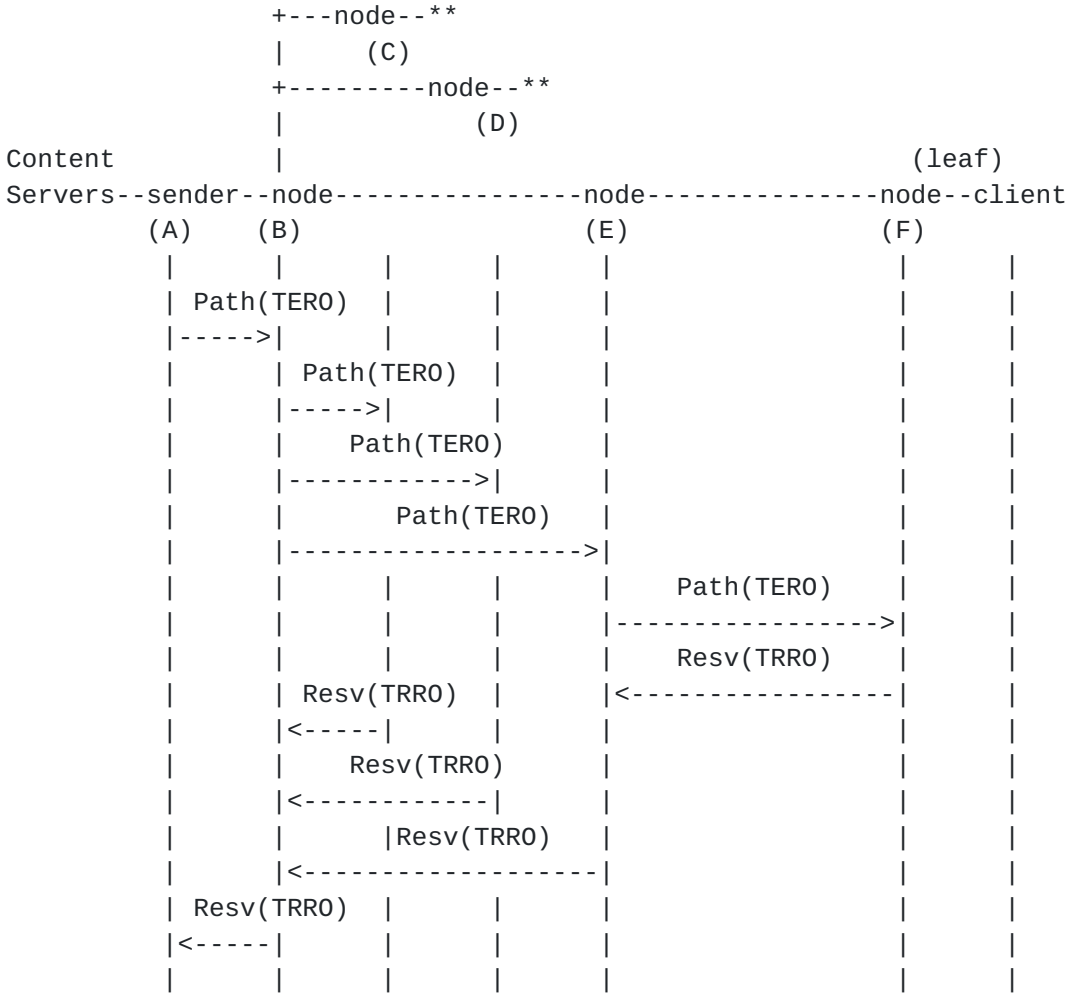


Figure 6 Sequence of sender-initiated multicast LSP establishment events

4.2 Processing of TERO and TRRO for sender-initiated multicast LSP establishment

A node that receives a Path message refers to the SESSION object and SENDER_TEMPLATE. The node determines whether the Path message is for an existing LSP or a new LSP.

If the Path message is for a new LSP, then the following procedure applies. The node verifies that the first subobject (hop) in the TERO is an address or interface belonging to this node. If not, then an appropriate error message MUST be sent upstream. The specific error message to send is described elsewhere in this document. The error to return is "Bad EXPLICIT_ROUTE object", as specified in RFC 3209 [1]. If the TERO contains only one hop, then this node is a leaf node and sends a Resv message upstream. Otherwise, the TERO is processed to locate all hops with a distance that is 1 greater than the distance value contained in the first TERO hop. For each such hop, a unicast Path message MUST be sent to the node specified by the TERO hop.

If the Path message is for an existing LSP, then the following procedure applies. If the first subobject in the TERO contains the same sequence number as in the previously received Path message, then the entire TERO is treated as if no change has occurred and processing ends. Otherwise, the TERO is processed as follows. Locate each hop in the TERO with a distance that is 1 greater than the distance value contained in the first TERO hop and add it to a next hop list. If there are any downstream nodes for which this node holds path state that do NOT appear in the next hop list, then a PathTear message MUST be sent immediately to that node. Then, for each hop in the next hop list, if there is no path state for this next hop, then a unicast Path message MUST be sent to the node specified by the TERO hop. Otherwise (path state is found), if the new sequence number is different than the sequence number previously recorded in the path state, then a unicast Path trigger message MUST be sent to the node specified by the TERO hop.

Note that the TERO in each outgoing Path message MUST contain just the subtree with the next hop node as the root node of that subtree.

Each node that receives a Resv message containing a LABEL object uses that label for outgoing traffic associated with this LSP tunnel. If the node is not the sender node, it allocates a new label and places that label in the corresponding LABEL object of the Resv message, which it sends upstream to the PHOP. A branch node that sends multiple Path messages to downstream nodes MUST forward a Resv message upstream ONLY after it receives a reply, whether positive (Resv) or negative (PathErr), from ALL downstream nodes to whom it sent a Path message. In order to handle non-reply from downstream nodes, a node MAY start a response timer upon sending outgoing PATH messages. If the response timer expires, then each outgoing PATH message for which no reply was received is treated the same as if a PathErr message had been received.

See section "Error Handling and Error Codes" for a description of

how to handle received PathErr messages.

When Resv messages are merged, a new TRRO is made using TRROs included in the received Resv messages. This new TRRO expresses the multicast subtree. The branch node that merges Resv messages is the root node of multicast subtree. The TRRO of the Resv messages is needed to detect loops on the multicast tree. The node SHOULD record information of TRRO.

4.3 Teardown mechanism

A multicast LSP tunnel is explicitly torn down by a PathTear message. The PathTear message must be routed exactly like the corresponding Path message. The PathTear message is copied by each branch node and is forwarded downstream to delete the corresponding path state. PathTear messages are initiated not only by the sender node but also by the branch node that receives a Path message with a TERO from which downstream nodes have been removed.

The process of pruning nodes from an existing multicast tree is described in section 5.3.

4.4 Path/Resv error

During the multicast LSP establishment described in section 4.1 and 4.2, various errors may occur. The main reasons are bandwidth allocation failures and unknown next hops specified by TERO. If a node does not support a new object or new C-type defined by this protocol, it sends error messages indicating "unknown object class" error or an "Unknown object C-Type". A node that initiates a ResvErr message SHOULD send ResvErr messages to all leaf nodes that are affected by the error.

4.5 Message format

4.5.1 Path message format

TERO is added to the Path message. Note that a new C-Type is added to SESSION object.

```
<Path Message> ::= <Common Header> [ <INTEGRITY> ]
                                    <SESSION>
                                    <RSVP_HOP>
                                    <TIME_VALUES>
                                    <TREE_EXPLICIT_ROUTE>
                                    <LABEL_REQUEST>
                                    [ <SESSION_ATTRIBUTE> ]
                                    [ <POLICY_DATA> ... ]
                                    <sender descriptor>

<sender descriptor> ::=  <SENDER_TEMPLATE> <SENDER_TSPEC>
                                        [ <ADSPEC> ]
                                        [ <RECORD_ROUTE> ]
```

4.5.2 Resv message format

TRRO is added to the Resv message. Note that a new C-Type is added to SESSION object.

```
<Resv Message> ::= <Common Header> [ <INTEGRITY> ]
                                   <SESSION><RSVP_HOP>
                                   <TIME_VALUES>
                                   [ <RESV_CONFIRM> ]  [ <SCOPE> ]
                                   [ <POLICY_DATA> ... ]
                                   <STYLE> <flow descriptor list>

<flow descriptor list> ::= <FF flow descriptor list>
                                      | <SE flow descriptor>

<FF flow descriptor list> ::= <FLOWSPEC> <FILTER_SPEC>
                                <LABEL> [ <TREE_RECORD_ROUTE> ]
                                        | <FF flow descriptor list>
                                <FF flow descriptor>

<FF flow descriptor> ::= [ <FLOWSPEC> ] <FILTER_SPEC>
                         <LABEL>[ <TREE_RECORD_ROUTE> ]

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

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

<SE filter spec> ::= <FILTER_SPEC> <LABEL> [ <TREE_RECORD_ROUTE> ]
```

4.5.3 Other RSVP message formats

With the exception of a new C-Type for the Session object, there is no change in the format of PathTear, PathErr, ResvErr and ResvConf.

5. Sender-initiated grafting/pruning mechanism

This section describes maintenance of an existing multicast tree. All setup and teardown of a multicast tree or subtree is controlled by the sender node. Changes to be made are signaled to downstream nodes by making changes in the outgoing TERO object. To add (graft) a subtree to a multicast tree, new hops are added to the TERO. To remove (prune) a subtree from an existing multicast tree, hops are removed from the TERO.

The basis for choosing the mechanism for determining that a change to the multicast tree is requested is that the TERO object can grow quite large as the number of levels and number of branches at each level increases. For instance, using all IPv4 hops, The TERO object for a tree with 5 levels and 3 branches at each node is about 2900 octets. With 5 levels and 5 branches per node, the TERO grows to about 31000 octets. If for each refresh Path message the entire TERO were compared for changes, the amount of time spent processing the Path message could be quite long.

With this in mind, the sequence field is included in each TERO hop subobject. When a tree topology change is required, the sender node changes the outgoing TERO and puts a new sequence number in each TERO hop along the series of hops towards the graft or prune node. This enables each downstream node to quickly determine whether the entire TERO needs to be checked for specific changes.

5.1 Sender-initiated grafting mechanism

The grafting mechanism supports grafting of a partial multicast tree (subtree) onto an established multicast tree. We define the node that performs subtree establishment as a graft node.

The sender node initiates grafting by sending an updated TERO in the outgoing Path message towards the graft node. Each TERO hop object along the path towards the graft node MUST have it's sequence number set to a value that is 1 greater than the value previously sent in the first hop object in the TERO.

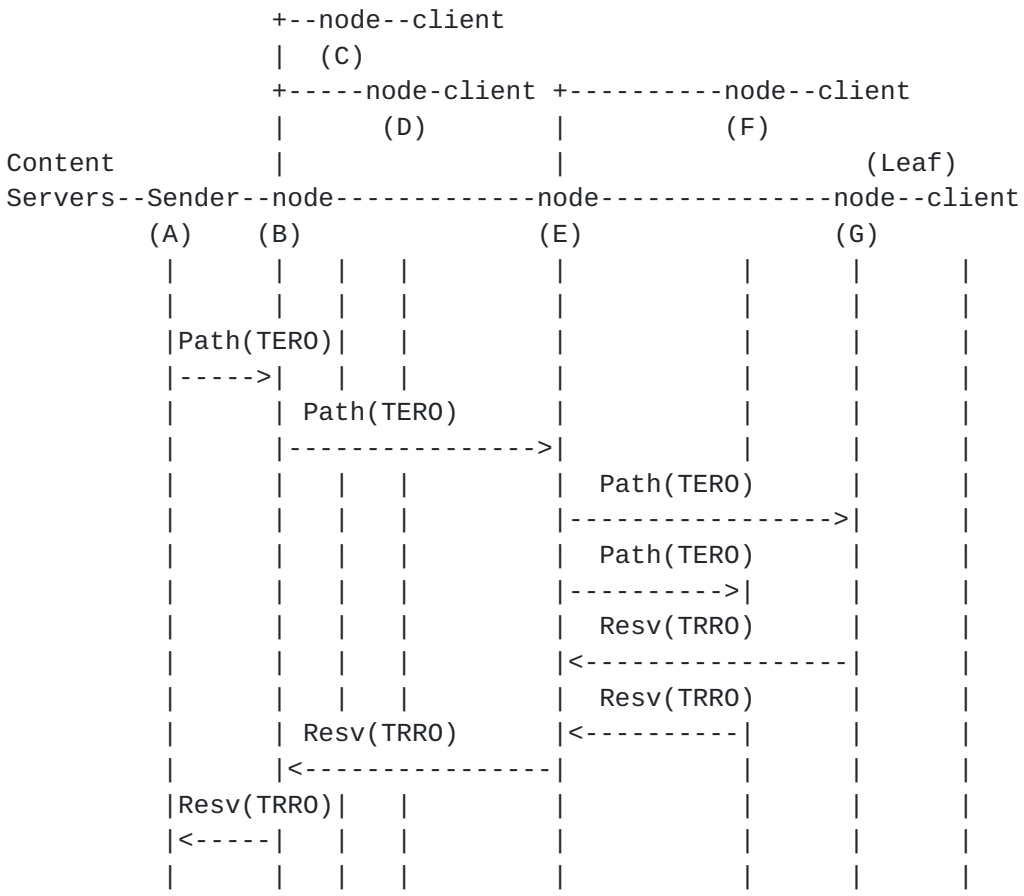
The processing of the TERO object at each node is as described in section "Processing of TERO and TRRO for sender-initiated multicast LSP establishment".

For example, refer to figure 4. Assume all nodes in the figure are members of the multicast tree except D & E, and that the current sequence number in all TERO hops is 23. Now, in order to graft nodes D & E into the tree, the following TERO would be sent by node A to node B:

{B(1,24),C(2,24),D(3,24),E(3,24),F(2,23),G(2,23),H(3,23)}

where N(d,s) means
Node = N
distance = d
sequence = s

Figure 7 shows the message sequence of the proposed grafting process.



Original TERO = {A(1),B(1),C(1),D(1)}
Modified TERO = {A(2),B(2),C(1),D(1),E(2),F(2),G(2)}
Sequence number in parentheses.

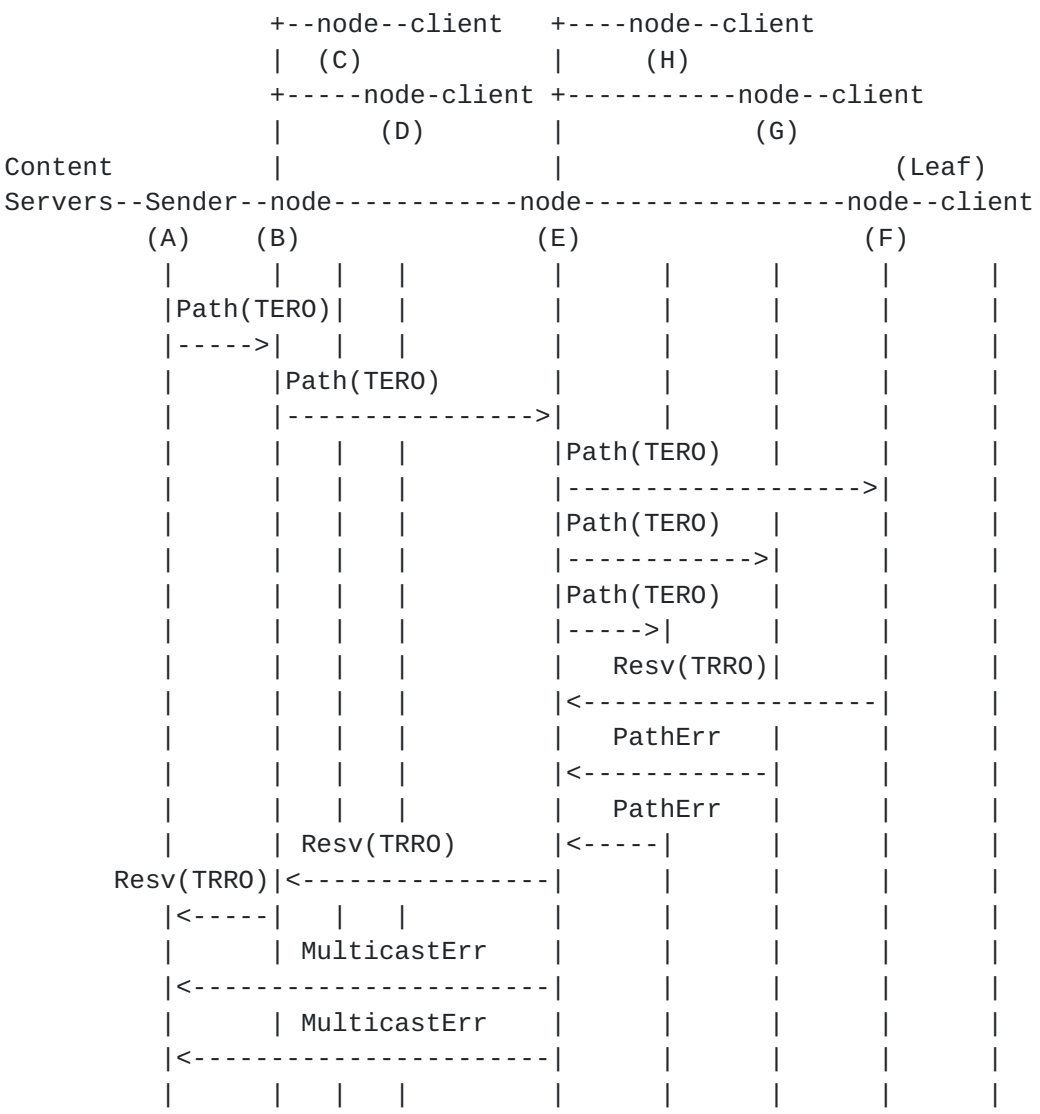
Figure 7: Sequence of graft process

5.2 Graft Error

Figure 8 shows the error message sequence of the grafting process. In this example, part of the expanded tree tunnel cannot be established for some reason, such as a resource error or label assignment error. Each node that cannot establish an LSP MUST send a PathErr message

that includes the error code to the graft node.

See section "Error Handling and Error Codes" for a description of how to handle received PathErr messages.



Original TERO = {A(1),B(1),C(1),D(1)}
Modified TERO = {A(2),B(2),C(1),D(1),E(2),F(2),G(2),H(2)}
Sequence number in parentheses.

Figure 8: Sequence of graft error process

A node supporting this multicast mechanism MUST support the graft mechanism described in this section. When a graft node that is requested to perform a grafting process detects some error in this message, it SHOULD send a MulticastErr message that includes the error code to the sender node. The following are

examples of errors:
- No route available toward destination (from RFC 3209 [1])
  a portion of or the whole subtree setup failed because no route was available

5.3 Sender-initiated pruning mechanism

The pruning mechanism supports pruning of unnecessary multicast subtree tunnels from an established multicast tree tunnel. It allows an intermediate node to prune unnecessary multicast subtree tunnels. We define the node that performs the pruning as a prune node.

The sender nodes initiates pruning by sending an updated TERO in the outgoing Path message towards the prune node. Each TERO hop object along the path towards the prune node MUST have it's sequence number set to a value that is 1 greater than the value previously sent in the first hop object in the TERO.

The processing of the TERO object at each node is as described in section "Processing of TERO and TRRO for sender-initiated multicast LSP establishment".

For example, refer to figure 4. Assume all nodes in the figure are members of the multicast tree, and that the current sequence number in all TERO hops is 23. Now, in order to prune node D from the tree, the following TERO would be sent by node A to node B:

```
{B(1,24),C(2,24),E(3,23),F(2,23),G(2,23),H(3,23)}

   where N(d,s) means
      Node = N
      distance = d
      sequence = s
```

Note that node D does not appear in the TERO. Once node C receives the TERO, node C sends a PathTear message to node D since path state is held by node C for node D.

Figure 9 shows the message sequence of the pruning process.

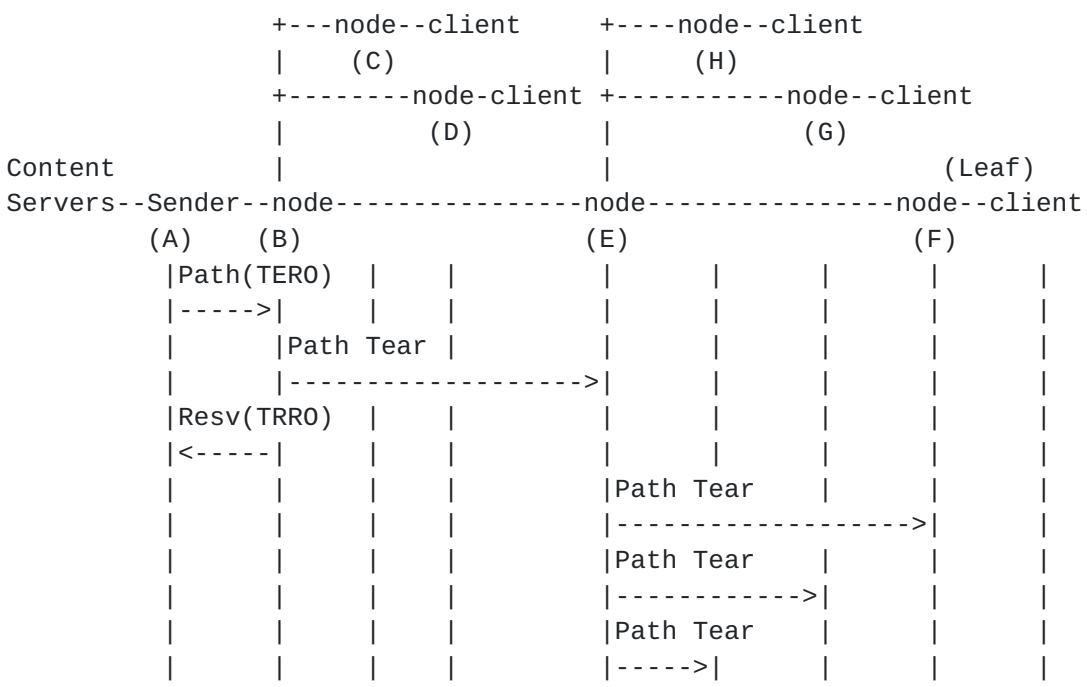


Figure 9: Sequence of prune process

6. Leaf-initiated multicast LSP establishment

6.1 Leaf-initiated joining mechanism

A leaf node wishing to join a multicast tree sends a Join message corresponding to the tree. Here, we call the node a join leaf node. A Join message is a trigger for establishing a QoS path requested by a join leaf node.

The Join message is addressed to the sender node. The Router Alert option MUST be disabled. Therefore, the Join message is sent as a hop-by-hop routed IP datagram until reaching the sender node.

A Join message has no specifications about the QoS parameters of the expanded path, i.e., there is no SenderTSpec object in the Join message. The sender node knows the QoS parameters because it has already included them in the Path message for setting up the tree. Therefore, the join leaf node inherits the same QoS parameters as every other node in the tree.

The process of expanded path establishment is as follows. The leaf

node wishing to join a multicast tree sends a Join message to the sender node. After the sender node receives the message, it checks that it is in fact the sender node for the specified tunnel. If not, it sends a JoinErr message to the join leaf node. Otherwise, the sender node calculates a path to reach the leaf node. If no such path can be calculated, a JoinErr message is sent to the leaf node. Otherwise, the sender node sends out an updated TERO in the outgoing Path message towards the joining leaf node. Each TERO hop object along the path towards the leaf node MUST have it's sequence number set to a value that is 1 greater than the value previously sent in the first hop object in the TERO.

The processing of the TERO object at each node is as described in section "Processing of TERO and TRRO for sender-initiated multicast LSP establishment".

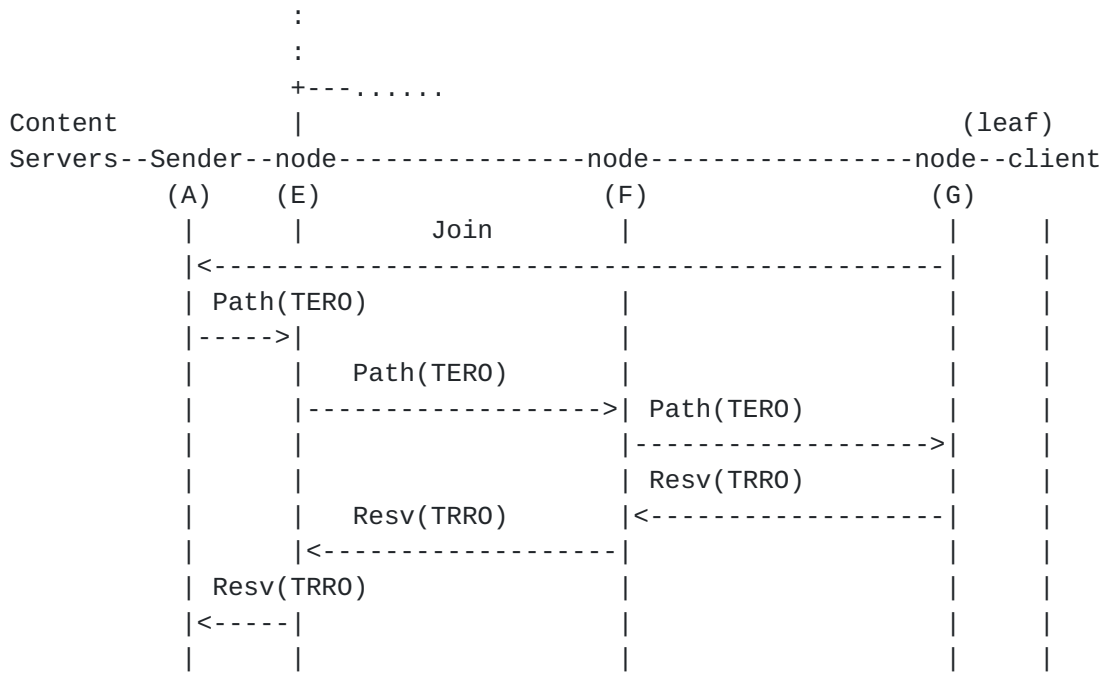


Figure 10: Sequence of join process

6.2 Join Error

During the Join process described in the previous section, various errors may occur. In the Join process, the join leaf node is notified of the errors and analyzes them. For this purpose, the JoinErr message is defined, which is sent by the sender node to the join leaf

node. It is sent when the graft node receives a PathErr message or detects an internal error. The JoinErr message is sent addressed to the leaf node with the Router Alert option disabled. The following are examples of errors:

- Node is not sender node of multicast tree
  The receiving node has is the sender node for the specified multicast tree
- No route available toward destination (from RFC 3209 [1])
  The subtree setup failed because no route was available

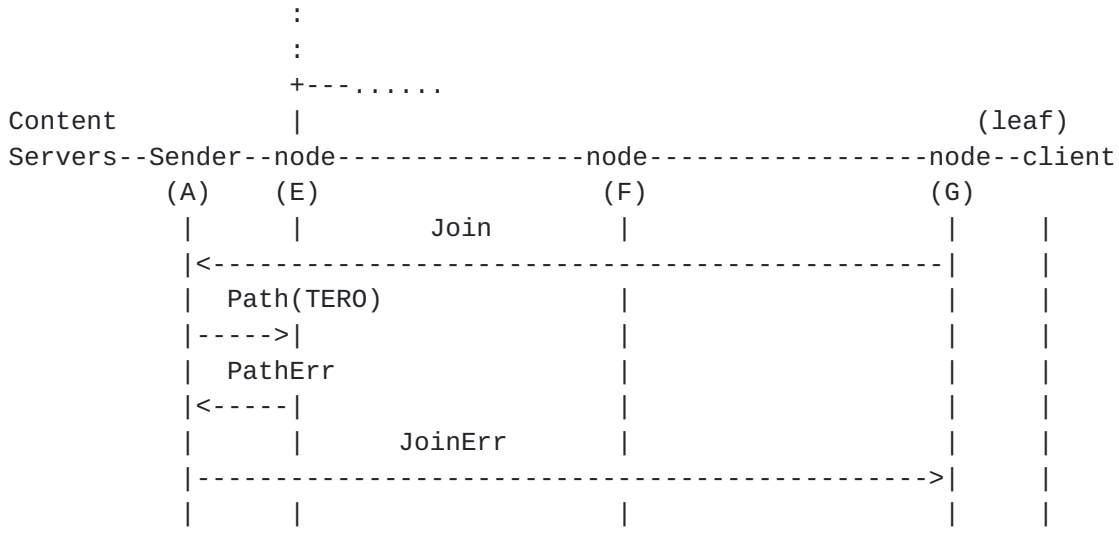


Figure 11: Sequence of JoinErr process

6.3 Leaf-initiated leaving mechanism

A leaf node wishing to leave a multicast tree sends a Leave message corresponding to the tree. Here, we call the node a leave leaf node. A Leave message is a trigger for tearing down a portion of a multicast tree.

The Leave message is addressed to the sender node. The Router Alert option MUST be disabled. Therefore, the Leave message is sent as a hop-by-hop routed IP datagram until reaching the sender node.

The process of leaving is as follows. The leaf node wishing to leave a multicast tree sends a Leave message to the sender node. After the sender node receives the message, it checks that it is in fact the sender node for the specified tunnel. If not, it sends a LeaveErr

message to the leave leaf node. Otherwise, the sender node removes the leave leaf node from the TERO. If the leave leaf node is not in the current TERO, a LeaveErr message is sent to the leaf node. Otherwise, the sender node sends out an updated TERO in the outgoing Path message towards the leave leaf node. Each TERO hop object along the path towards the leaf node MUST have it's sequence number set to a value that is 1 greater than the value previously sent in the first hop object in the TERO.

The processing of the TERO object at each node is as described in section "Processing of TERO and TRRO for sender-initiated multicast LSP establishment".

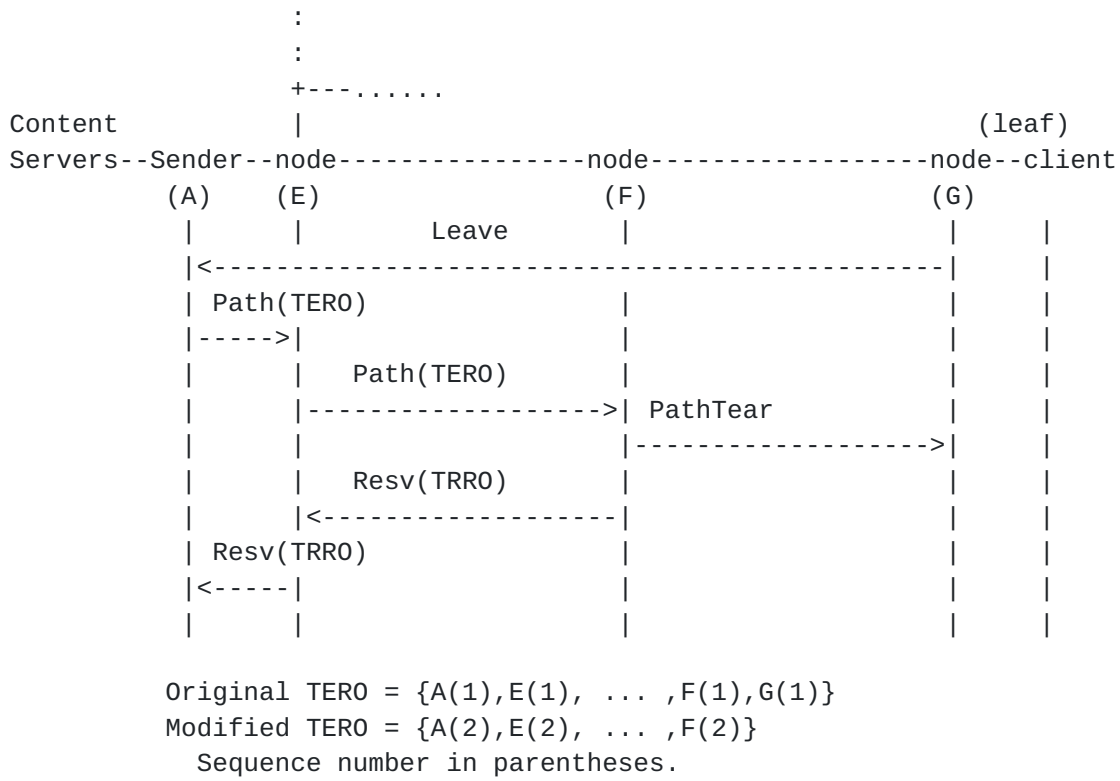
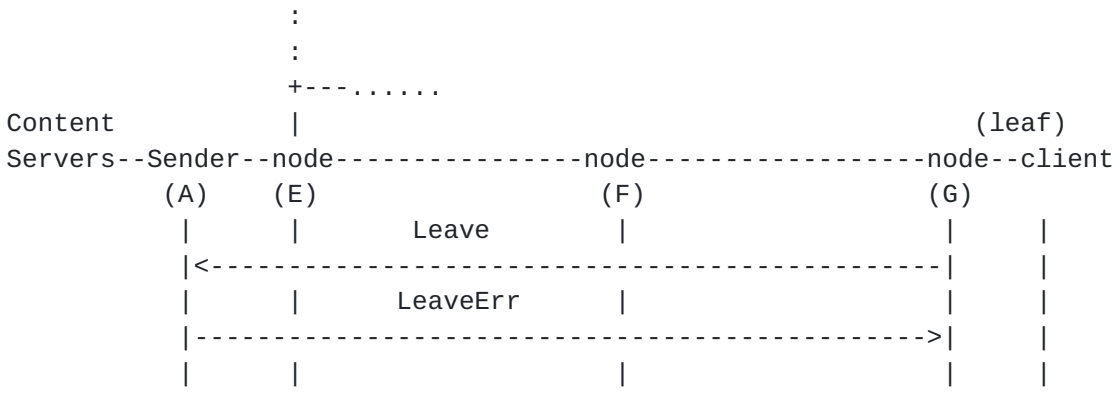


Figure 12: Sequence of leave process

6.4 Leave Error

During the leave process described in the previous section, various errors may occur. In the leave process, the leave leaf node is notified of the errors and analyzes them. For this purpose, the LeaveErr message is defined, which is sent by the sender node to the leave leaf node. It is sent when the sender node receives an error message or detects an internal error. The LeaveErr message is sent addressed to the leaf node with the Router Alert option disabled. The following are examples of errors:

- Node is not sender node of multicast tree
  The receiving node is not the sender node for the specified multicast tree
- Leaf node not found in TERO
  The sender node could not locate the leaf node in the TERO



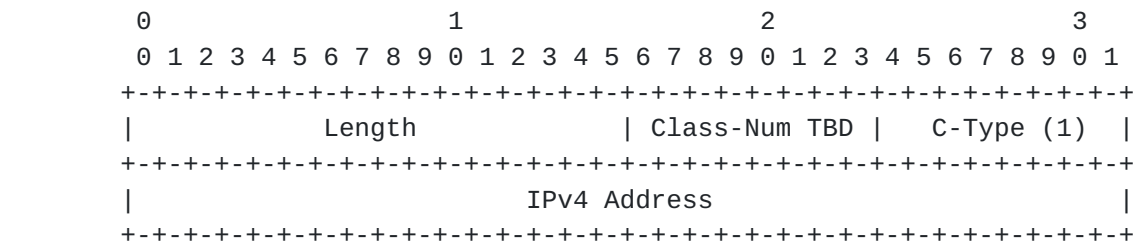
Original TERO = {A(1),E(1), ... ,F(1)}
Note that G is not a member of the tree.
Sequence number in parentheses.

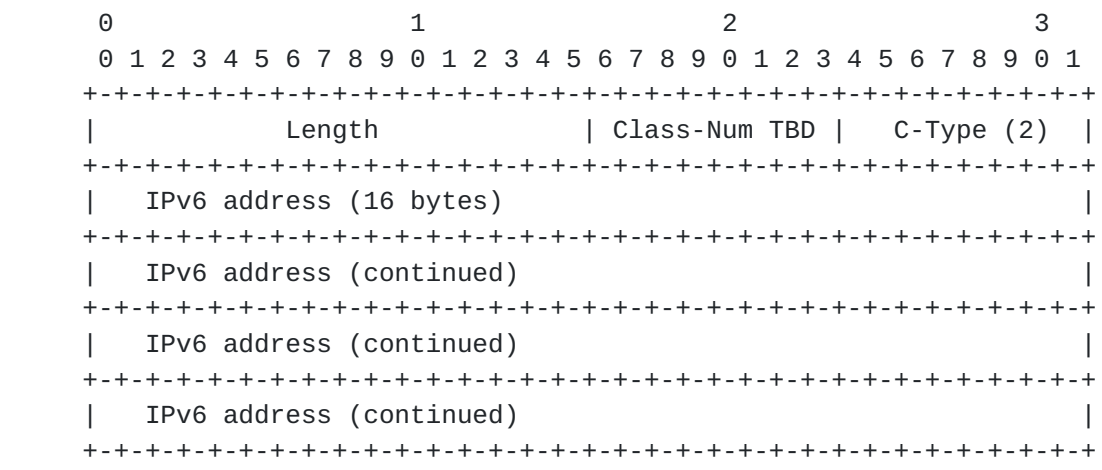
Figure 13: Sequence of LeaveErr process

6.5 Message formats

The following sections describe the formats for messages used during leaf-initiated join and leave operation. A new object, LEAF_NODE_ID, is introduced to tell upstream nodes the identity of the requesting leaf node. The format of the LEAF_NODE_ID object is as follows:

Class = LEAF_NODE_ID





Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

Class-Num

The Class Number for the LEAF_NODE_ID object. TBD.

C-Type

0x01 IPv4 address
0x02 IPv6 address

IPv4/IPv6 Address

An IPv4 or IPv6 address identifying the leaf node requesting the Join or Leave message.

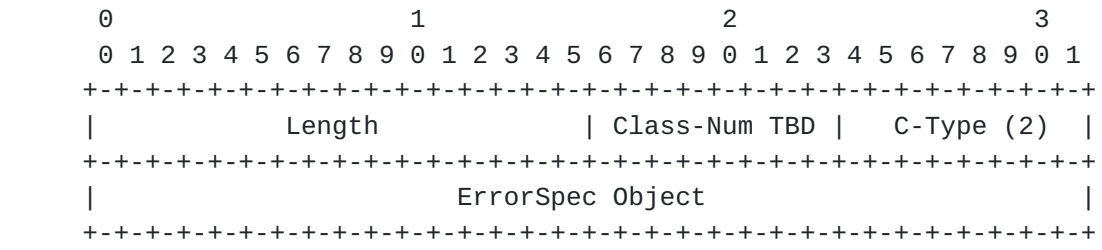
6.5.1 Join message format

```
<Join Message>::=<Common Header>
                 <SESSION>
                 <LEAF_NODE_ID>
                 <sender descriptor>

<sender descriptor>::=(see earlier definition)
```

6.5.2 JoinErr message format

The JoinErr message utilizes the MulticastNotify message with a MULTICAST_NOTIFY object as follows:



Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

Class-Num

The Class Number for the MULTICAST_NOTIFY object. TBD.

C-Type

2 = JoinErr

ErrorSpec Object

The ErrorSpec object either copied from the received PathErr message or locally generated.

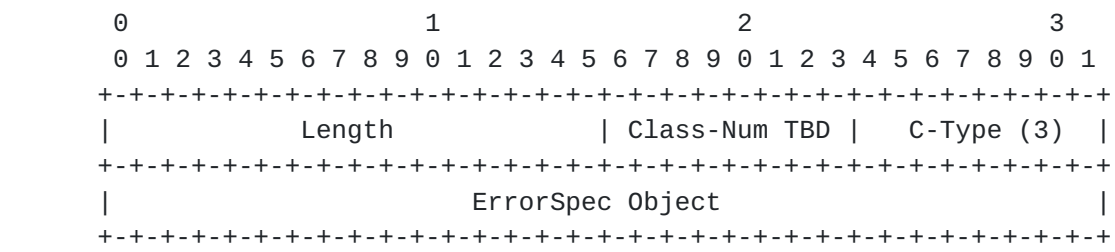
6.5.3 Leave message format

```
<Leave Message>::=<Common Header>
                  <SESSION>
                  <LEAF_NODE_ID>
                  <sender descriptor>

<sender descriptor>::=(see earlier definition)
```

6.5.4 LeaveErr message format

The LeaveErr message utilizes the MulticastNotify message with a MULTICAST_NOTIFY object as follows:



Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

Class-Num

The Class Number for the MULTICAST_NOTIFY object. TBD.

C-Type

3 = LeaveErr

ErrorSpec Object

The ErrorSpec object either copied from the received PathErr message or locally generated.

7. Error Handling and Error Codes

This section describes error handling and the values to use for newly introduced errors.

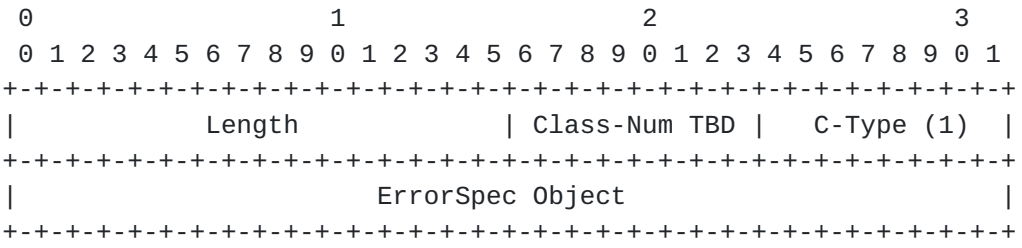
The handling of received PathErr and Resv messages at a branch node and at a non-branch node is described in more detail. Note that the following descriptions assume that ALL downstream nodes have responded, or that the response timer has expired.

7.1 Error Handling at Branch Node

If no part of the subtree setup succeeded, (i.e., no Resv messages were received), then a single PathErr message with error "Subtree setup completely failed" MUST be sent to the PHOP node.

Otherwise (i.e., at least one Resv message was received), a Resv message MUST be sent to the PHOP node. The TRRO object is merged from all the received TRRO objects, after which the current node adds itself to the TRRO. Then, for each received PathErr message, a MulticastErr message containing the ErrorSpec object copied from the PathErr message MUST be sent to the sender node with the Router Alert option disabled.

The MulticastErr message utilizes the MulticastNotify message with a MULTICAST_NOTIFY object as follows:



Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields.

Class-Num

The Class Number for the MULTICAST_NOTIFY object. TBD.

C-Type

1 = MulticastErr

ErrorSpec Object

The ErrorSpec object either copied from the received PathErr message or locally generated.

7.2 Error Handling at Non-Branch Node

If a PathErr message is received, it MUST be sent to the PHOP without altering the ErrorSpec object. Otherwise, a Resv was received and a Resv MUST be sent to the PHOP node. The TRRO object is generated by adding the current node to the TRRO from the received Resv.

7.3 Error During Resv Processing

A node detecting an error during Resv processing (e.g., traffic control failure) MUST treat this as if a PathErr message had been

received from all affected downstream nodes. Then, the processing follows the procedures in other subsections of this section.

A node receiving a Resv message with the style object specifying a different style than that requested in the Path message MUST treat this as an error and proceed as above in this section.

7.4 Error Codes and Error Value Sub-Codes

The following list extends the basic list of Error Codes and Values that are defined in [RFC2205].

Error Code	Meaning
TBD	Multicast Problem
	This Error Code has the following globally-defined Error Value sub-codes:
	1 Node is not sender node of multicast tree
	2 Leaf node not found in TERO
	3 Subtree setup completely failed

8. Use of Refresh Reduction

The multicast extensions in this document introduce several new messages. These messages are paramount to the successful signaling and maintenance of multicast trees. These messages are sometimes transmitted to non-adjacent nodes and are not refreshed in the manner that Path and Resv messages are. If one of these messages is dropped in transit, it will then never reach the intended destination node. It is therefore important to ensure that these messages are successfully received by the intended destination node.

In order to achieve this goal, timers with retransmission should be used. The refresh reduction extension [12] to RSVP already defines such a mechanism.

Therefore, it is RECOMMENDED that implementations of this multicast extension also implement and enable refresh reduction with message identifiers and ack_desired.

9. Application for Traffic Engineering

9.1 Rerouting Traffic Engineered Multicast Tunnels

Traffic Engineering supports rerouting of an established TE tunnel. The route is decided according to the administrative policy. The detection of a more optimal path and network resource failure are examples of situations where path rerouting is needed.

While TE tunnel rerouting is in progress, an important requirement is avoiding traffic disruption. A useful solution to this problem is the make-before-break concept. In this concept, traffic passes through the old path while the new rerouted path is being established. Once it has been established, traffic is switched to the new path and the old path is torn down. The make-before-break concept supports resource sharing of the common part where the old and new path cross. This sharing is useful to avoid competition between the resources of the two paths. Details of the make-before-break concept are given in RFC 3209. In the case of multicasting, there are cases where the topology of the area to be rerouted is a tree and there are many paths that should be rerouted when path rerouting occurs at once. So it is desirable that multiple paths are moved in one rerouting mechanism.

This protocol supports the make-before-break concept for multicast TE tunnels. For this version of this protocol, only sender-initiated make-before-break of the entire multicast tree is supported. Local repair using make-before-break is a topic for future study.

9.2 Re-establishment of subtree

The graft mechanism can also be used re-establish a partial multicast LSP when the establishment of the subtree failed. In multicast LSP establishment, it sometimes happens that the establishment of the subtree fails and the establishment of another part of the tree succeeds. It is inefficient in this situation for the whole multicast LSP to be torn down by the PathTear message and for the whole multicast LSP to be re-established by the Path message. Re-establishment of a subtree whose establishment failed can be done using the graft mechanism. The sender node or NMS that recognizes that the establishment of a subtree failed can re-calculate another multicast route that avoids the failure point and can reach the leaf nodes. After this re-calculation, the sender node uses the graft mechanism with updated TERO for subtree re-establishment. This re-establishment does not cause any additional processing in nodes except along the path to the failure point, because the graft process only affects those nodes.

10. Security Considerations

Security considerations will be addressed in a future revision of

this document.

11. References

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

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

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

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

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

[6] W. Fenner, "Internet Group Management Protocol, IGMP, version 2", RFC 2236, November 1997.

[7] Cain, B., "Internet Group Management Protocol, Version 3", draft-ietf-idmr-igmp-v3-11.txt, May 2002

[8] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification.", RFC 2362, June 1998.

[9] J. Moy, "Multicast Extensions to OSPF", RFC 1584, March 1994

[10] Lou Berger, et al., "Generalized MPLS - Signaling Functional Description", draft-ietf-mpls-generalized-signaling-08.txt, April 2002

[11] Kireeti Kompella, Yakov Rekhter, "Signalling Unnumbered Links in RSVP-TE", draft-ietf-mpls-rsvp-unnum-07.txt, July 2002.

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

[13] Eric Mannie ,

"Generalized Multi-Protocol Label Switching (GMPLS) Architecture", draft-ietf-ccamp-gmpls-architecture-03.txt, August 2002

[14] D. Katz, D. Yeung, K. Kompella, "Traffic Engineering Extensions to OSPF Version 2", draft-katz-yeung-ospf-traffic-08.txt, September 2002

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

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

12. Author's Addresses

Seisho Yasukawa
NTT Network Service Systems Laboratories, NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 4769
EMail: yasukawa.seisho@lab.ntt.co.jp

Masanori Uga
NTT Network Service Systems Laboratories, NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 4804
EMail: uga.masanori@lab.ntt.co.jp

Hisashi Kojima
NTT Network Service Systems Laboratories, NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 6070
EMail: kojima.hisashi@lab.ntt.co.jp

Koji Sugisono
NTT Network Service Systems Laboratories, NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-Shi, Tokyo 180-8585 Japan
Phone: +81 422 59 2605
EMail: sugisono.koji@lab.ntt.co.jp

Alan Kullberg
Netplane Systems
8 Technology Drive
Westborough, MA
EMail: akullber@netplane.com

Markus Jork
Avici Systems
101 Billerica Avenue
N. Billerica, MA 01862
phone: +1 978 964 2142
EMail: mjork@avici.com

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