DECLARATION

I, Laura Nugent, based on my personal knowledge and information, hereby declare as follows:

1. I am Director of Administration and Events of the IETF Administration LLC ("IETF"). IETF is the acronym for the Internet Engineering Task Force, which is an activity of the Internet Society.

2. One of my responsibilities with IETF is to act as the custodian of Internet-Drafts and records relating to Internet-Drafts. I am familiar with the record keeping practices relating to Internet-Drafts, including the creation and maintenance of such records.

3. I hereby declare that all statements made herein are of my own knowledge and information contained in the business records of IETF and are true, and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements may be punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

4. If depositions regarding the information in this declaration are required, the deposition should be taken by phone or videoconference or, if it must be in person, should be in California.

5. Since 1998, it has been the regular practice of the IETF to publish Internet-Drafts and make them available to the public on its website at www.ietf.org (the IETF website). The IETF maintains copies of Internet-Drafts in the ordinary course of its regularly conducted activities.

6. Any Internet-Draft published on the IETF website was reasonably accessible to the public and was disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence could have located it. In particular, the Internet-Drafts were indexed and searchable on the IETF website.

7. Internet-Drafts are posted to an IETF online directory. When an Internet-Draft is published, an announcement of its publication that describes the Internet-Draft is disseminated. Typically, that dated announcement is made within 24 hours of the publication of the Internet-Draft. The announcement is kept in the IETF email archive and the date is affixed automatically.

8. The records of posting the Internet-Drafts in the IETF online repository are kept in the course of the IETF's regularly conducted activity and ordinary course of business. The records are made pursuant to established procedures and are relied upon by the IETF in the performance of its functions.

9. It is the regular practice of the IETF to make and keep the records in the online repository, housed with a cloud vendor who operates under contract with the IETF Administration LLC.

10. Exhibit 1 is a true and correct copy of draft-asveren-dime-state-recovery-02.txt, titled "Diameter State Recovery Considerations." I have determined that an announcement of the publication of this Internet-Draft was made on December 11, 2007. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

11. Exhibit 2 is a true and correct copy of draft-bodin-dime-auditing-reqs-03.txt, titled "Auditing Functionality in Diameter." I have determined that an announcement of the publication of this Internet-Draft was made on September 10, 2007. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

12. Exhibit 3 is a true and correct copy of draft-ietf-idr-bgp4-10.txt, titled "A Border Gateway Protocol 4 (BGP-4)." I have determined that an announcement of the publication of this Internet-Draft was made in May 2, 2000. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

13. Exhibit 4 is a true and correct copy of draft-ietf-isis-traffic-01.txt, titled "IS-IS extensions for Traffic Engineering." I have determined that an announcement of the publication of this Internet-Draft was made in June 29, 1999. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

14. Exhibit 5 is a true and correct copy of draft-ietf-mpls-rsvp-lsp-tunnel-05.txt, titled "RSVP-TE: Extensions to RSVP for LSP Tunnels." I have determined that an announcement of

the publication of this Internet-Draft was made in March 10, 2000. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

15. Exhibit 6 is a true and correct copy of draft-ietf-mpls-rsvp-lsp-tunnel-07.txt, titled "RSVP-TE: Extensions to RSVP for LSP Tunnels." I have determined that an announcement of the publication of this Internet-Draft was made in August 25, 2000. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

16. Exhibit 7 is a true and correct copy of draft-ietf-mpls-rsvp-lsp-tunnel-08.txt, titled "RSVP-TE Extension to RSVP for LSP Tunnels." I have determined that an announcement of the publication of this Internet-Draft was made in February 28, 2001. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

17. Exhibit 8 is a true and correct copy of draft-katz-yeung-ospf-traffic--09.txt, titled "Traffic Engineering Extensions to OSPF Version 2." I have determined that an announcement of the publication of this Internet-Draft was made in October 24, 2002. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24

hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

18. Exhibit 9 is a true and correct copy of draft-schelen-nsis-opopsig-00.txt, titled "On path and off path signaling for NSIS." I have determined that an announcement of the publication of this Internet-Draft was made in June 28, 2002. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

19. Exhibit 10 is a true and correct copy of draft-schelen-nsis-opopsig-01.txt, titled "Path-coupled and Path-decoupled Signaling for NSIS." I have determined that an announcement of the publication of this Internet-Draft was made on November 8, 2002. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

Pursuant to Section 1746 of Title 28 of United States Code, I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that the foregoing is based upon personal knowledge and information and is believed to be true.

Date: 23 May 2024 By: _____
Laura Nugent

4890-8440-2593

Network Working Group T. Asveren
Internet-Draft Sonus Networks
Expires: June 13, 2008 U. Bodin

Operax

December 11, 2007

Diameter State Recovery Considerations
draft-asveren-dime-state-recovery-02.txt

Status of this Memo

Copyright Notice

Abstract

This document discusses parameters to consider, different approaches
and design strategies to synchronize and/or recover state in Diameter
applications after failure of an active instance.

Table of Contents

1. Introduction

There are a variaety of Diameter applications defined to perform different tasks. For some of these tasks, synchronizing and/or recovering state for ongoing sessions after failure of a Diameter endpoint is desirable, e.g. Diameter Credit Control Application. The recovery could be achieved by a proprietary mechanism, could be assisted by protocol mechanisms or could be a combination thereof. This document focuses on issues associated with protocol assisted state recovery.

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 [2].

The following terms defines the functionality used in describing entities in this document.

Ongoing Session

A Diameter session, for which at least the first transaction has been completed but not the last transaction according to the application message flow.

Terminated Session

A Diameter session that existed in the past, for which the last transaction according to the application message flow has been completed.

Initial message

A Diameter message used to create a new Diameter session.

Mid-session message

A Diameter message used to refresh or modify an existing Diameter session.

Service Instance

An instance of service provided by a Diameter application to another entity, e.g. charging, authentication services.

Diameter Transaction

A Diameter request/answer pair.

3. Session State and the Need for Recovery

Some Diameter applications make use of sessions consisting of multiple transactions. The context necessary to be able to process/ trigger further messages in an ongoing session constitutes the session state.

In multi-transaction sessions, it is possible that one of the endpoints fail during a session. Depending on the application, it may not be possible/desirable to terminate the corresponding service instance. In such a case, it is necessary to utilize a backup node which can process messages for the ongoing session or to use a new session without terminating the service instance.

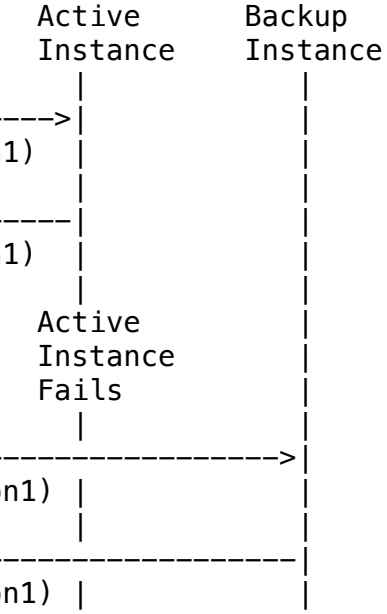


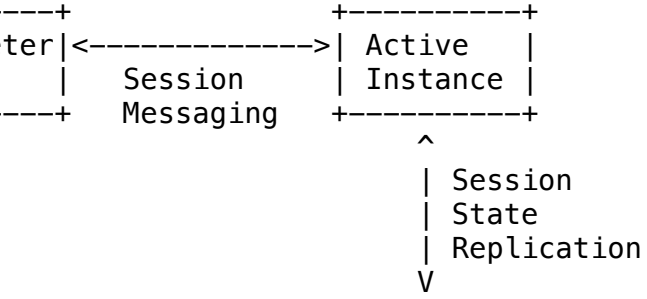
Figure 1: Session Failover to Backup Instance

Another important aspect related with failing instances is the possibility of hanging resources on the peer Diameter entity. This could happen if the peer Diameter entity does not clean up session state unless the session is terminated according to the expected application message flow. It should be noted that while state recovery is a desirable feature for certain applications, hanging

resources is an unacceptable situation for all applications, hence although some of the mechanisms described in this document could be used to prevent the occurance of such a case, it is recommended that application layer mechanisms, e.g. application layer timers, are used for this purpose. Nonetheless, certain strategies mentioned in this document could be used to expedite session state cleanup after failovers.

4. Proprietary Mechanisms

Proprietary mechanisms do not assume any specific behavior from their peers. They usually rely on some form of state replication between active and backup instances.



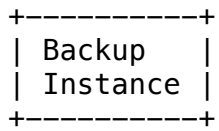


Figure 2: Data Replication with a Proprietary Machanism

It should be noted that Figure 2 is just an abstract representation of proprietary data replication between active and backup instances.
Actual implementation may vary depending on the mechanims used. Proprietary state synchronization is a common technique utilized by Public Switched Telephone Network equipment vendors to provide 5 9's reliability. There are also initiatives to define a standard set of APIs for platforms/middleware providing data synchronization services, e.g. Application Interface Specification of Service Availability Forum.

Proprietary data replication between active and backup instances may be asynchronous in nature. This means that they may not provide loss-less state replication at all times. Hence, after a failover to a backup instance, some session states may have been lost and other states may be wrongly kept by the backup instance. That is, states may have been terminated through session signalling to the initially

active instance but the removal of the corresponding session states were not properly reflected in the data replication process.

5. Protocol Assisted State Recovery

Protocol assisted state recovery relies on contents of the messages exchanged between Diameter entities.

5.1. Service Models

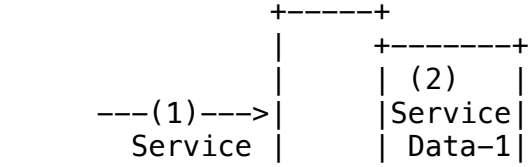
For each Diameter session Diameter messaging happens between a client and server. Although not a sender/receiver of Diameter messages, physical service/resource provided is also a parameter when designing state recovery mechanisms. The physical resource/service is application dependent and could be bandwith allocated on a router for QoS application, voice transfer resources used for a prepaid voice call application etc.

Depending on Diameter application, physical resource/service could be at the client or server side. For example for Diameter Credit Control Application the physical resource is controlled by the client, whereas for QoS application with a push scenario it is controlled by the server.

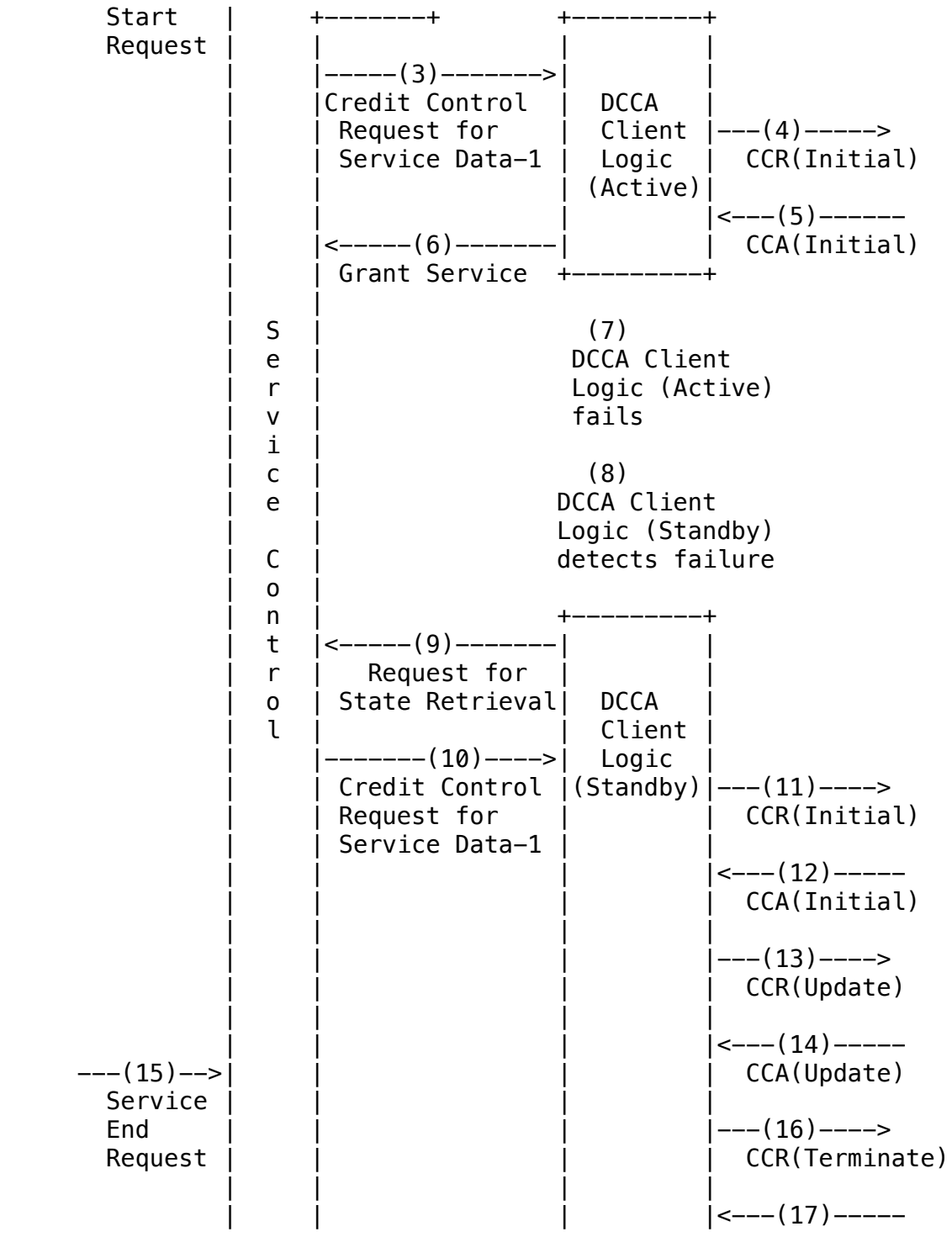
In case a proprietary data replication mechanism which is not loss-less is used between active and backup instances to support failover, it may be desirable to make use of the data present in the physical resource/service. This case can benefit from a synchronization phase before session data is transfered for purposes of rebuilding lost state.

Physical resource/service could be used to extract some information regarding session state to be reconstructed. For certain scenarios this information could be enough for state reconstruction or could be used in addition to information obtained via other means, e.g. in a proprietary data replication mechanism, failovers could be followed by a synchronization phase based on information obtained from the physical resource/service.

Below is given a conceptual diagram for the DCCA client side state recovery utilizing the state kept by service control logic.



6]


Start Request
Credit Control Request for Service Data-1
DCCA Client Logic (Active)
Grant Service
Service Control
(7) DCCA Client Logic (Active) fails
(8) DCCA Client Logic (Standby) detects failure
Request for State Retrieval
DCCA Client Logic (Standby)
Credit Control Request for Service Data-1
CCR(Initial)
CCA(Initial)
CCR(Update)
CCA(Update)
CCR(Terminate)
Service End Request

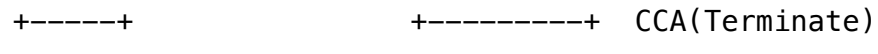


Figure 3: Using Service Information for DCCA Client Side State Recovery

5.2. Parameters to Consider

There are several aspects which may be important for a protocol assisted session state recovery mechanism. They may or may not be part of the design choices for a protocol assisted session state recovery mechanism, depending on the strategy utilized.

5.2.1. Notification of the Peer About Failure

Usually it is necessary for the remote peer to be informed about the failure of the active instance in the context of protocol assisted state recovery. This could be achieved in different ways:

Application Layer Timers

Application layer timers could be utilized to send new requests periodically. Lack of a new request or a corresponding answer for a sent request/receipt or UNABLE_TO_DELIVER error answer could indicate that the peer Diameter entity has failed.

Notification from Standby Instance

After failure of the active instance, standby instance can send a message to the remote Diameter peer to inform it about failure of the active instance. This method requires standby instance to know the identities of the remote Diameter peers, with which the failed active instance had ongoing sessions. This information could be exchanged by a proprietary data replication mechanism. Alternatively, standby instance could have a configured list of remote peers and notify all of them.

5.2.2. Transfer of Session Data

For protocol assisted recovery it is necessary to supply enough information to the backup instance so that session state can be constructed. What constitutes session state data needs to be defined on a per application basis. Also, in certain cases (e.g. when a separate mechanism for state replication is used in combination with protocol assisted state recovery) the transfer of session data may be preceeded by a state synchronization phase. For example, a generic message providing a list of all active sessions could be used for such a synchronization phase.

Some approaches to transfer session data include:

Using a New Session

Upon detection of the failure of the active instance, remote Diameter peer may start a new session without terminating the service instance.

Using Application Messages

Data necessary to reconstruct the session state may be transferred in an application defined message by AVP(s) specifically defined for that purpose. Alternatively, an AVP may be used to flag that all data carried in the message is sent for the purposes of state synchronization.

Using a Generic Message

Data necesary to reconstruct session state may be transferred in a message specifically defined for that purpose. Such a message may carry state information for one or multiple sessions.

5.2.3. Backup Server Selection

A Diameter peer needs to know the identity of the backup instance, so that it can send the necessary data to reconstruct session state. Furthermore, loadbalancing of the ongoing sessions to different backup instances may be necessary as well, to prevent overloading of backup entities.

Active Instance Guided Selection

Active instance could communicate the identity of the backup instance(s) to the peer Diameter entity with an AVP. Information about how the load should be distributed among multiple backup instances could be communicated as well.

Backup Instance Guided Selection

If the notification of the peer Diameter entity about the failure of the active instance is performed via a message sent by the standby instance, the identity of the backup instance would be known to the the peer Diameter entity. This message could carry information about other backup instances and loadsharing information too.

Selection Based on Configuration

The Diameter peer may know the identities of backup servers through configuration and try to loadshare ongoing session based

on a locally defined algorithm. For requests, which are rejected by a standby instance with TOO_BUSY_HERE error answer, another standby instance could be tried.

5.2.4. Timing of State Reconstruction

When state reconstruction should happen may vary depending on the application. The following two models are foreseen:

State Reconstruction After Failure

It may be necessary to reconstruct the state after the backup instance detects failure of the active instance. This model is useful when the state for ongoing sessions is necessary to generate answers for requests belonging to new sessions. Care should be taken when determining the necessary information for such cases, it could be the case that what is needed is some cumulative data based on session states rather than the per session information and this could impact the design choices to recover/replicate the data or even the choice between a proprietary mechanism and protocol assisted recovery.

Another use case is when autonomous requests need to be generated from the side, where the active instance has failed. In such a situation, backup instance needs to know ongoing sessions immediately after it detects failure of the active instance so that it can generate such requests.

If state reconstruction after failure is needed, notification of the Diameter peer about failure should be done by the backup instance.

State Reconstruction Upon Receipt of a Request

For certain applications, it could be enough if a backup server can reply for requests for ongoing sessions after the failure of the active instance. In such scenarios, state information contained in the new requests for ongoing sessions (i.e. mid-session messages) could be used to reconstruct session state on the standby instance.

5.3. Approaches

The choice between a proprietary and protocol assisted state recovery mechanism is not a straightforward one. Depending on the application and the reliability level required a detailed analysis needs to be

done to justify usage of one of the methods.

If it is desired to use protocol assisted recovery, parameters discussed in Section 5.2 need to be considered. It should be noted that choices made for different parameters are not always independent of each other, e.g. if state reconstruction immediately after failure detection is necessary, using a new session to transfer session data strategy can't be utilized. Below, two different approaches are discussed in detail.

5.3.1. Using a New Session

As mentioned in Section 5.2.2 a new session can be used to rebuild state after failure. This approach can be sufficient if immediate state reconstruction after failure is not needed. That is, knowledge of the history of the session are not needed to proceed providing the service of the failed over Diameter node. An example diagram is given in Figure 3. It focuses on events happening on the client side for a DCCA session. On the server side, the sessions which were created by the active instance are cleaned up after expiry of Tcc timer.

A variant of using a new session for rebuilding state is to use application messages. For example, regular mid-session messages maintaining soft-state can be used if they contain enough information for the desired state reconstruction. Such messages could contain an AVP carrying a flag indicating that it's a mid-session message and not an initial message issued to create a completely new session. The ability to separate between recreated session and new session can be important to some applications. For example, it may be

desirable
to give recreated sessions preference over new session to resources controlled by a Diameter server.

5.3.2. Backup Instance Triggered Recovery

In case immediate state reconstruction is desired or strictly needed by a backup Diameter instance, this instance may need to trigger transfer of session data to recover state. This requires session data to be available and reachable to the backup Diameter instance. Possible locations of such data include the physical resource/service controlled by the failed over Diameter instance and the entities utilizing the service offered by the Diameter instance (i.e. entities issuing Diameter requests for the offered service).

As mentioned in Section 5.2.2 application application messages or a generic message can be used to transfer session data for state reconstruction. Application messages or a generic message transferring the desired session data could be preceeded by a generic synchronization message providing the backup Diameter instance with a complete list of all active sessions. By that the backup Diameter

instance can distribute the recovery of session data over time. This may be useful if this instance is to start provide its service imediately instead of waiting until the state reconstruction process is completed. Requesting session data in parallel with answering to service requests requires however that period with incomplete session state after that the backup Diameter instance starts providing the service is acceptable.

A generic synchronization message can also be useful in a combined

solution using both a proprietary mechanism for state replication and protocol aided state recovery. The complete list of all active sessions provided in such a message providing can be compared with the list of sessions replicated through a proprietary mechansism. Thereby a potential mis-match can be identified and missing session data can be explicitly requested by the backup Diameter instance.

6. IANA Considerations

This document does not require any IANA action.

7. Security Considerations

Certain procedures in protocol assisted state recovery, e.g. notification of the Diameter peer about failure of an active instance by the standby instance, could introduce security risks. It is expected that use of IPSec/TLS together with a transitive trust model should eliminate these concerns.

8. Acknowledgments

9. Normative References

[1] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

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

Authors' Addresses

Tolga Asveren
Sonus Networks
4400 Route 9 South
Freehold, NJ, 07728
USA

Email: tasveren@sonusnet.com

Ulf Bodin
Operax
Aurorum Science Park 8
SE-977 75 Lulea
Sweden

Email: uffe@operax.com

Full Copyright Statement

Intellectual Property

assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

Network Working Group U. Bodin
Internet-Draft Operax
Intended status: Standards Track A. Doria (ed.)
Expires: March 2, 2008 Lulea University of Technology
B. Chatras
France Telecom
S. Norreys
BT
August 30, 2007

Auditing Functionality in Diameter
draft-bodin-dime-auditing-reqs-03.txt

Status of this Memo

This Internet-Draft will expire on March 2, 2008.

Copyright Notice

Abstract

Diameter is being increasingly included in the work of other standards organizations and has become a key protocol in many architectures. One of the uses of Diameter includes setting and

maintaining hard-state and soft-state during failover and in the event of delayed refresh messages respectively. Often there is a need to query for information on active sessions for backup or synchronization purposes.

Table of Contents

1. Terminology and Conventions

The key words MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Introduction

Diameter has been widely adopted as a base protocol for different interfaces of next generation network (NGN) architectures developed by 3GPP, ETSI TISPAN and the ITU-T. Some of these interfaces are used to support hard state as well as to support soft state. For example, in the ETSI TISPAN NGN architecture the service policy decision function (SPDF) offers a Diameter based interface facing application functions over which they can issue resource reservation requests for various media flows. Such an application function can, e.g., be a SIP based soft-switch or a portal for media streaming. The interface between the SPDF and applications functions must support both hard and soft state.

This contribution offers three failover use cases, two for hard-state and one for soft-state, that would benefit from the addition of an auditing function to Diameter. The primary requirement being set out in this draft is the requirement for retrieving state for failover and synchornization purposes.

2.1. Definitions of hard and soft state reservations

In this draft, hard-state reservations and soft-state reservation

will be used in the meanings documented by ETSI TISPAN[ETSI06].

- Hard-state reservation: type of reservation whereby the requested resources are reserved without time limit. Hard-state reservations are terminated if the DIAMETER session is terminated.
- Soft-state reservation: type of reservation whereby the requested resources are reserved for a finite amount of time. Soft-state reservations are terminated when the DIAMETER session is terminated.

2.2. Replication

The replication of session data can be performed regularly to facilitate instantiation of fresh server platform using replicated data. This approach is herein referred to as replication for warm standby. Another approach is to replicate session data in real-time to a parallel server platform, which immediately can take over the primary server's tasks in case it fails or becomes unavailable. This

approach is referred to as replication for hot standby. The case when no session data is replicated, a fresh server platform taking over the primary server's tasks is referred to as a cold standby.

It should be noted that refresh messages may be desirable not only for soft-state reservations. By having clients refreshing their active sessions before a timeout in the server it can be ensured that the clients remain in sync with the server. When a timeout occurs at the absence of an expected refresh message the server would, in this scenario, keep the session and issue a callback to the client informing it that a session exists that has not been refreshed accordingly. The client may then either provide a refresh or explicitly terminate the session by replying to this callback.

The behavior of issuing a callback to the client when a session timeout occurs can clearly be useful also for soft-state sessions. For example, clients failing in refreshing sessions would with such callbacks not need to re-establish sessions that should not have been allowed to expire and handle possible consequences of session data being wrongly removed. Refresh failures may for example occur if a delayed refresh arrives after a session has already been deleted. However, in contrst to when hard-state is maintained, a soft-state server needs to eventually terminate a session which is not being refreshed in time (e.g. after a short period following a callback).

Issuing a callback to ask whether a session is still active is generally a useful mechanism to assure that clients and their server remain syncronized. That is, a client or a server can benefit from the ability to issue such a question triggered by any internal or external event to make sure a particular session (or set of sessions) is still active in the peer node. For example, in case refresh messages are not used for a hard-state server, the Diameter server could issue a callback for sessions that have been active for a longer period to make sure they are still active in the corresponding client.

2.3. Diameter used for hard-state reservations

There are at least two common use cases when Diameter is used for hard-state reservations:

- without replication
- with replication

2.3.1. In the case of failover without replication

In cases where hard state is used over a Diameter interface in an environment where nodes have backups in case of failure, client nodes

need a mechanism to audit their server for active sessions. That is, in case a Diameter client node crashes, its backup needs to audit the server node for active sessions. Otherwise the backup node cannot know which states are active and can't terminate them when they are no longer needed.

These cases show that an auditing mechanism is needed to support hard-state whenever session information is replicated for resilience purposes. It is also clear that the auditing mechanisms needs to be symmetrical in order to support both the client auditing for session information in the server and the server auditing the client.

2.3.2. In the case of failover with replication

A Diameter client, server, or both may replicate session state information over several database instances at different nodes to facilitate seamless node failovers. Replication of data over several database instances are often done asynchronously to keep response times low. That is, with asynchronous replication (i.e., unacknowledged database writes) a Diameter server can answer immediately to a client request instead of waiting for data to be properly replicated before answering. When using hard-state Diameter clients and their server face the risk of getting out of sync after a failover.

As a consequence of asynchronous replication, session state requested and established in a Diameter server node may not have been properly replicated before the server crashes and is seamlessly replaced by its backup (e.g., through IP takeover or SCTP multi-homing). The server may, however, have responded to the request before crashing. The Diameter client could, therefore, record that lost (hard) session state is still active in the server when it is not. On the other hand, in case the client is terminating an active session and the server fails in replicating the state removal before crashing the backup server node will maintain a hard session state of which the client is unaware and which is invalid.

2.4. Diameter used for soft-state reservations

In the case of soft-state reservations, an auditing mechanism can still be beneficial at failover between servers and between clients.

At failover between servers the backup server taking over can request clients to re-establish their sessions. Instead waiting for refresh messages to arrive is likely to increase the time needed to get in sync. This is particularly true when long timeout periods are used.

At failover between clients that do not replicate data (i.e., cold standby), an auditing mechanism would may allow a backup client to

re-establish at least part of the original session data. In case all or part of the data possible to obtain from the server is useless to the backup client, it can choose to explicitly terminate the corresponding sessions. This would shorten the period at which clients and their server are out of sync after a failover.

2.5. Required Mechanisms

The cases outlined above require that auditing mechanisms support both queries for a list of active sessions and support specific queries for detailed session information kept by the queried node (i.e., either the client or the server node).

A further requirement for an auditing mechanism is that it must be able to work in parallel with the basic runtime operation of the Diameter signaling and interrupt that signaling. For example, in case a large number of audit messages is needed to synchronize, the rate of those messages must be possible to limit leaving enough capacity to the basic runtime signaling to handle ordinary session setup and teardown.

Support for queries to check if a particular session or a set (list)

of sessions are still active in the peer node is also required (i.e.
either the server or a client node).

3. Diameter Resource Management Extensions

The contents of this section are based on draft-calhoun-diameter-res-mgmt-08.txt [id-res-mgmt] which was submitted in March 2001 by Pat R. Calhoun. The recommendation of the authors of this draft is that we use this solution as the starting point for meeting the requirements listed above.

3.1. Extension Overview

Diameter [RFC3588] is an authentication, authorization and accounting (AAA) protocol used for network access services, such as dial-up (NASREQ) [RFC4005] and Mobile IP [id-framework] .

The NASREQ AAA requirements [RFC3169] require that AAA servers maintain session state information. This is typically used to enfore a local policy decision, such as limiting the number of simultaneous sessions for a specific user, maintaining IP address pools, etc. The AAA WG's network access requirements [RFC2989] require that an AAA protocol be able to query for session state information, in the event that this information is lost.

This extension describes an extension to the Diameter protocol that allows a Diameter node to query for active session state information from its peers in order to rebuild state information. Although it is envisioned that this would be used when state information was lost,

and needed to be rebuilt, it is possible for a node to periodically query for state information in order to ensure that its state is current.

This document only concerns itself with the ability to query for session state information. Resources are actually reserved when a user is successfully authorized. Therefore, relevant application-specific extensions, such as [RFC4005] and [id-framework] , MUST define what resources are to be managed, by specifying what AVPs MUST be present in the Resource-Token AVP.

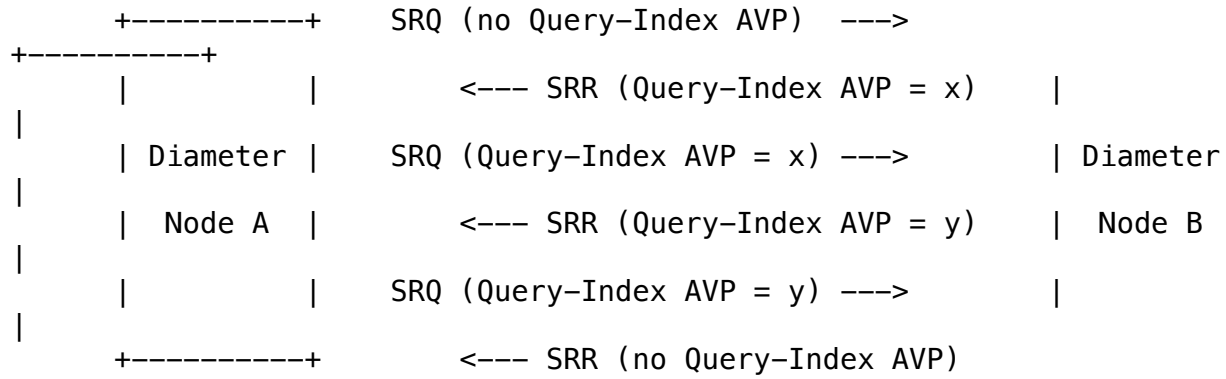
The Extension number for this draft is three (3). Diameter nodes conforming to this specification MUST include an Extension-Id AVP with a value of three in the Device-Reboot-Ind Command [RFC3588] .

3.1.1. State synchronization

When a Diameter node determines that it is has lost all state information it had for a specific peer, it SHOULD issue a Session-Resource-Query message to the peer. The node in question MAY postpone all authorization messages from the peer until state has been restored.

Upon receipt of the Session-Resource-Query, all Resource-Token AVPs for the requested sessions, indicated via one or more Session-Id AVP, MUST be returned in a Session-Resource-Reply. The absence of any Session-Id AVP is an indication that all active sessions are to be returned.

If the node is unable to send all of the information within a single message, it MUST include the Query-Index AVP, with a value that has local significance. A node that receives a Session-Resource-Reply with a Query-Index AVP SHOULD issue another Session-Resource-Query message with the Query-Index AVP intact, requesting the rest of the state information.



The above example depicts Diameter Node a issuing an SRQ to Node B. Upon replying with an SRR, node B determines that it is unable to include all of the Resource-Token AVPs in a single reply, and therefore includes the Query-Index AVP with a value of x.

Upon receipt of the response, node A processes all Resource-Token AVPs and issues a subsequent SRQ with the Query-Index AVP set to x. Node B receives the SRQ, and using the Query-Index AVP determines which sessions need to be included in the corresponding SRR.

This exchange continues until node B returns an SRR that does not include the Query-Index AVP, indicating that there is no further session state information to be returned.

3.2. Command-Code Values

This section defines Command-Code [RFC3588] values that MUST be supported by all Diameter implementations conforming to this specification. The following Command Codes are defined in this specification:

Command Name	Abbrev.	Code	Reference
Session-Resource-Query	SRQ	277	Section 3.2.1
Session-Resource-Reply	SRR	278	Section 3.2.2

3.2.1. Session-Resource-Query (SRQ)

The Session-Resource-Query (SRQ), indicated by the Command-Code field set to 277, MAY be sent by a Diameter node to any of its peer to request a state update. The presence of one or more Session-Id AVPs in the Session-Resource-Query message indicates that the server

only
   wants to receive the Resource-Token for the specified session(s).

   Message Format

```
   <Session-Resource-Query>  ::= < Diameter Header: 277 >
                                      { Extension-Id }
                                      { Origin-FQDN }
                                      { Origin-Realm }
                                      { Destination-Realm }
                                    * [ Session-Id ]
                                    * [ Destination-FQDN ]
                                   0*1[ Query-Index ]
                                    * [ AVP ]
                                    * [ Proxy-State ]
                                    * [ Route-Record ]
                                    * [ Routing-Realm ]
                                   0*1< Integrity-Check-Value >
```

3.2.2. Session-Resource-Reply (SRR)

   The Session-Resource-Reply (SRR), indicated by the Command-Code field
   set to 278, is sent in response to a SRQ message. The SRR message
   contains a Resource-Token for each active session that was requested
   via the Session-Id AVP. The absence of any Session-Id AVP in the

SRQ
   implies that Resource-Tokens for all active sessions MUST be
   returned.

   In the event that all of the state information cannot be sent at
   once, the SRR message MUST include the Query-Index AVP.

```
      <Session-Resource-Reply>  ::= < Diameter Header: 278 >
                                      { Extension-Id }
                                      { Origin-FQDN }
                                      { Origin-Realm }
                                      { Result-Code }
                                      [ Destination-FQDN ]
                                   0*1[ Query-Index ]
                                    * [ Resource-Token ]
                                    * [ AVP ]
                                    * [ Proxy-State ]
                                    * [ Route-Record ]
                                    * [ Routing-Realm ]
                                   0*1< Integrity-Check-Value >
```

3.3. Mandatory AVPs

   The following table describes the Diameter AVPs defined in the
   Resource Management extension, their AVP Code values, types,
possible
   flag values and whether the AVP MAY be encrypted.

Attribute Name	AVP Code	Section Defined	Data Type	AVP Flag rules: MUST	MAY	SHLD NOT	MUST NOT	Encr
Query-Index	500	4.4.1	Unsigned32	M	P		V	Y

```
|
   Resource-Bag     502  4.4.3   OctetString| M  |  P  |    |  V  | Y
|
   Resource-Token   501  4.4.2   Grouped    | M  |  P  |    |  V  | Y
|
```

3.3.1. Query-Index AVP

The Query-Index AVP (AVP Code 500) is of type Unsigned32 and MUST only be present in the Session-Resource-Query and the Session-Resource-Reply messages. The Query-Index AVP has local significance to the issuer of the Session-Resource-Reply message, and is used to identify the state information that remains to be sent in a subsequent SRR message.

3.3.2. Resource-Token AVP

The Resource-Token AVP (AVP Code 501) is of type Grouped. The value is a set of AVPs used to track state information that is pertinent to an active session. The issuer of the SRR message is responsible for creating a Resource-Token AVP for all active sessions requested.

The following describes the minimum number of AVPs that MUST be present in a Resource-Token AVP. Service-specific AVPs MAY also be present, as defined in the appropriate service extension document.

```
resource-token = sid host user timestamp extension-id optional
sid            = Session-Id AVP ; See [RFC3588] , Section 3.1.
host           = Host-Name AVP ; See [RFC3588] , Section 2.3.1.
user           = User-Name AVP ; See [RFC3588] , Section 3.3.
timestamp      = Timestamp AVP ; See [RFC3588] , Section 7.3.
extension-id   = Extension-Id AVP ; See [RFC3588] , Section 2.6.3.
optional       = Resource-Bag AVP ; See Section 3.3
```

The Host-Name AVP contains the NAI of the access router that is servicing the user, while the timestamp AVP contains the time at which the successful Diameter authorization response was received, and the service was initiated.

3.3.3. Resource-Bag AVP

This AVP allows encapsulation of arbitrarily many AVPs to be included in a Resource-Token. These AVPs are defined in service specific extensions to Diameter. The only restrictions to the AVPs is that

they MUST NOT be interpreted so as to conflict with the other fields of the Resource-Token Group value, namely, the Session-Id, Host-Name, User-Name, Timestamp or Extension-Id AVPs.

The Resource-Bag AVP (AVP Code = 502) is of type OctetString. The AVP encapsulates an arbitrary of AVPs, each with its own header and value.

3.4. Original IANA Considerations

The command codes defined in Section 2.0 are values taken from the Command-Code [RFC3588] address space and extended in [RFC4005] , [RFC4004] and [id-acct-ext] . IANA should record the values as defined in Section 3.2

The AVPs defined in section 3.0 were alllocated from from the AVP numbering space defined in [RFC3588] , and extended in [RFC4005] , [RFC4004] and [id-acct-ext] . IANA should record the values as defined in Section 3.3.

3.5. Original Security Considerations

This Diameter extension assumes that the Resource Management data is secured either through a hop-by-hop authentication mechanism, as described in [RFC3588] , or using a strong authentication mechanism as defined in [id-crypto-ext].

4. References

4.1. Normative References

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

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September

2003.

[RFC4004] Calhoun, P., Johansson, T., Perkins, C., Hiller, T., and P. McCann, "Diameter Mobile IPv4 Application", RFC 4004, August 2005.

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.

4.2. Informational References

[ETSI06] ETSI TISPAN, "Telecommunications and Internet converged Services and Protocols for Advanced Networking", 2006.

[RFC2989] Aboba, B., Calhoun, P., Glass, S., Hiller, T., McCann, P., Shiino, H., Zorn, G., Dommety, G., C.Perkins, B.Patil, D.Mitton, S.Manning, M.Beadles, P.Walsh, X.Chen, S.Sivalingham, A.Hameed, M.Munson, S.Jacobs, B.Lim, B.Hirschman, R.Hsu, Y.Xu, E.Campell, S.Baba, and E.Jaques, "Criteria for Evaluating AAA Protocols for Network Access", RFC 2989, November 2000.

[RFC3169] Beadles, M. and D. Mitton, "Criteria for Evaluating Network Access Server Protocols", RFC 3169, September 2001.

[id-acct-ext] Arkko, J., Calhoun, P., Patel, P., and G. Zorn, "Diameter Accounting Extensions", draft-calhoun-diameter-accounting-08.txt (work in progress), 2000.

[id-crypto-ext]

Calhoun, P., Bulley, W., and S. Farrell, "Diameter Strong
Security Extension",
draft-calhoun-diameter-strong-crypto-05.txt (work in
progress), 2000.

[id-framework]
Calhoun, P., Zorn, G., Pan, and Akhtar, "Diameter
Framework", draft-calhoun-diameter-framework-07.txt (work
in progress), 2000.

[id-res-mgmt]
"Diameter - Resource Management Extensions",
draft-calhoun-diameter-res-mgmt-08.txt (work in progress),
2001.

Appendix A. Changes

This section to be removed if and when this I-D is approved for publication as an RFC.

A.1. Changes in -03

1. Integrated draft-calhoun-diameter-res-mgmt-08.txt into the document
2. Updated references

A.2. Changes in -02

1. Added contents of draft-calhoun-diameter-res-mgmt-08.txt
2. Added TOC

A.3. changes in -01

1. Add some specificity on the purpose of the requirements
2. Added a soft state use case for synchronization.
3. Added a section on replication
4. Added informational reference to ETSI ES 283 026 with definitions
of soft and hard state

Appendix B. IANA considerations

TBD

Appendix C. Acknowledgements

The authors and editors of this specification thank Pat Calhoun for allowing us to use, include and buld on his proposed Diameter Resource Mnagement Extension draft.

C.1. Original Acknowledgements

The authors wish to thank Erik Guttman for providing some very useful
proposed text to handle the change in data types.

Authors' Addresses

Ulf Bodin
Operax
Lulea S-977 75
Sweden

Email: Ulf.Bodin@operax.com
URI: www.operax.com

Avri Doria

Lulea University of Technology
Lulea
Sweden

Email: avri@ltu.se
URI: psg.com/~avri

Bruno Chatras
France Telecom

Email: bruno.chatras@orange-ft.com
URI:

Steve Norreys
BT

Email: steve.norreys@bt.com
URI:

Full Copyright Statement

Intellectual Property

http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

Network Working Group
INTERNET DRAFT

Y. Rekhter
cisco Systems
T. Li
Procket Networks, Inc.
Editors

A Border Gateway Protocol 4 (BGP-4)
<draft-ietf-idr-bgp4-10.txt>

Status of this Memo

1. Acknowledgments

This document was originally published as RFC 1267 in October 1991, jointly authored by Kirk Lougheed and Yakov Rekhter.

We would like to express our thanks to Guy Almes, Len Bosack, and Jeffrey C. Honig for their contributions to the earlier version of this document.

We like to explicitly thank Bob Braden for the review of the earlier

version of this document as well as his constructive and valuable comments.

We would also like to thank Bob Hinden, Director for Routing of the Internet Engineering Steering Group, and the team of reviewers he assembled to review the previous version (BGP-2) of this document. This team, consisting of Deborah Estrin, Milo Medin, John Moy, Radia Perlman, Martha Steenstrup, Mike St. Johns, and Paul Tsuchiya, acted with a strong combination of toughness, professionalism, and courtesy.

This updated version of the document is the product of the IETF IDR Working Group with Yakov Rekhter and Tony Li as editors. Certain sections of the document borrowed heavily from IDRP [7], which is the OSI counterpart of BGP. For this credit should be given to the ANSI X3S3.3 group chaired by Lyman Chapin and to Charles Kunzinger who was the IDRP editor within that group. We would also like to thank Mike Craren, Dimitry Haskin, John Krawczyk, David LeRoy, John Scudder, John Stewart III, Dave Thaler, Paul Traina, and Curtis Villamizar for their comments.

We would like to specially acknowledge numerous contributions by Dennis Ferguson.

2. Introduction

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. It is built on experience gained with EGP as defined in RFC 904 [1] and EGP usage in the NSFNET Backbone as described in RFC 1092 [2] and RFC 1093 [3].

The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems (ASs) that reachability information traverses. This information is sufficient to construct a graph of AS connectivity from which routing loops may be pruned and some policy decisions at the AS level may be enforced.

BGP-4 provides a new set of mechanisms for supporting classless interdomain routing. These mechanisms include support for advertising an IP prefix and eliminates the concept of network "class" within BGP. BGP-4 also introduces mechanisms which allow aggregation of routes, including aggregation of AS paths. These changes provide support for the proposed supernetting scheme [8, 9].

To characterize the set of policy decisions that can be enforced using BGP, one must focus on the rule that a BGP speaker advertise to its peers (other BGP speakers which it communicates with) in neighboring ASs only those routes that it itself uses. This rule

reflects the "hop-by-hop" routing paradigm generally used throughout the current Internet. Note that some policies cannot be supported by the "hop-by-hop" routing paradigm and thus require techniques such as source routing to enforce. For example, BGP does not enable one AS to send traffic to a neighboring AS intending that the traffic take a different route from that taken by traffic originating in the neighboring AS. On the other hand, BGP can support any policy conforming to the "hop-by-hop" routing paradigm. Since the current Internet uses only the "hop-by-hop" routing paradigm and since BGP

can support any policy that conforms to that paradigm, BGP is highly
applicable as an inter-AS routing protocol for the current Internet.

A more complete discussion of what policies can and cannot be enforced with BGP is outside the scope of this document (but refer to the companion document discussing BGP usage [5]).

BGP runs over a reliable transport protocol. This eliminates the need to implement explicit update fragmentation, retransmission, acknowledgment, and sequencing. Any authentication scheme used by the transport protocol may be used in addition to BGP's own authentication mechanisms. The error notification mechanism used in BGP assumes that the transport protocol supports a "graceful" close, i.e., that all outstanding data will be delivered before the connection is closed.

BGP uses TCP [4] as its transport protocol. TCP meets BGP's transport requirements and is present in virtually all commercial routers and hosts. In the following descriptions the phrase "transport protocol connection" can be understood to refer to a TCP connection. BGP uses TCP port 179 for establishing its connections.

This document uses the term `Autonomous System' (AS) throughout. The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs. Since this classic definition was developed, it has become common for a single AS to use several interior gateway protocols and sometimes several sets of metrics within an AS. The use of the term Autonomous System here stresses the fact that, even when multiple IGPs and metrics are used, the administration of an AS appears to other ASs to have a single coherent interior routing plan and presents a consistent picture of what destinations are reachable through it.

The planned use of BGP in the Internet environment, including such issues as topology, the interaction between BGP and IGPs, and the enforcement of routing policy rules is presented in a companion

document [5]. This document is the first of a series of documents

planned to explore various aspects of BGP application. Please send comments to the BGP mailing list (bgp@ans.net).

3. Summary of Operation

Two systems form a transport protocol connection between one another. They exchange messages to open and confirm the connection parameters. The initial data flow is the entire BGP routing table. Incremental updates are sent as the routing tables change. BGP does not require periodic refresh of the entire BGP routing table. Therefore, a BGP speaker must retain the current version of the entire BGP routing tables of all of its peers for the duration of the connection. KEEPALIVE messages are sent periodically to ensure the liveness of the connection. NOTIFICATION messages are sent in response to errors or special conditions. If a connection encounters an error condition, a NOTIFICATION message is sent and the connection is closed.

The hosts executing the Border Gateway Protocol need not be routers. A non-routing host could exchange routing information with routers via EGP or even an interior routing protocol. That non-routing host could then use BGP to exchange routing information with a border router in another Autonomous System. The implications and applications of this architecture are for further study.

Connections between BGP speakers of different ASs are referred to as "external" links. BGP connections between BGP speakers within the

same AS are referred to as "internal" links. Similarly, a peer in a different AS is referred to as an external peer, while a peer in the same AS may be described as an internal peer. Internal BGP and external BGP are commonly abbreviated IBGP and EBGP.

If a particular AS has multiple BGP speakers and is providing transit service for other ASs, then care must be taken to ensure a consistent view of routing within the AS. A consistent view of the interior routes of the AS is provided by the interior routing protocol. A consistent view of the routes exterior to the AS can be provided by having all BGP speakers within the AS maintain direct IBGP connections with each other. Alternately the interior routing protocol can pass BGP information among routers within an AS, taking care not to lose BGP attributes that will be needed by EBGP speakers if transit connectivity is being provided. For the purpose of discussion, it is assumed that BGP information is passed within an AS using IBGP. Care must be taken to ensure that the interior routers have all been updated with transit information before the EBGP speakers announce to other ASs that transit service is being provided.

3.1 Routes: Advertisement and Storage

For purposes of this protocol a route is defined as a unit of information that pairs a destination with the attributes of a path to that destination:

- Routes are advertised between a pair of BGP speakers in UPDATE messages: the destination is the systems whose IP addresses are

reported in the Network Layer Reachability Information (NLRI) field, and the the path is the information reported in the path attributes fields of the same UPDATE message.

- Routes are stored in the Routing Information Bases (RIBs): namely, the Adj-RIBs-In, the Loc-RIB, and the Adj-RIBs-Out. Routes that will be advertised to other BGP speakers must be present in the Adj-RIB-Out; routes that will be used by the local BGP speaker must be present in the Loc-RIB, and the next hop for each of these routes must be present in the local BGP speaker's forwarding information base; and routes that are received from other BGP speakers are present in the Adj-RIBs-In.

If a BGP speaker chooses to advertise the route, it may add to or modify the path attributes of the route before advertising it to a peer.

BGP provides mechanisms by which a BGP speaker can inform its peer that a previously advertised route is no longer available for use. There are three methods by which a given BGP speaker can indicate that a route has been withdrawn from service:

a) the IP prefix that expresses destinations for a previously advertised route can be advertised in the WITHDRAWN ROUTES field in the UPDATE message, thus marking the associated route as being no longer available for use

b) a replacement route with the same Network Layer Reachability Information can be advertised, or

c) the BGP speaker - BGP speaker connection can be closed, which implicitly removes from service all routes which the pair of speakers had advertised to each other.

3.2 Routing Information Bases

The Routing Information Base (RIB) within a BGP speaker consists of three distinct parts:

a) Adj-RIBs-In: The Adj-RIBs-In store routing information that has been learned from inbound UPDATE messages. Their contents represent routes that are available as an input to the Decision Process.

b) Loc-RIB: The Loc-RIB contains the local routing information that the BGP speaker has selected by applying its local policies to the routing information contained in its Adj-RIBs-In.

c) Adj-RIBs-Out: The Adj-RIBs-Out store the information that the local BGP speaker has selected for advertisement to its peers. The routing information stored in the Adj-RIBs-Out will be carried in the local BGP speaker's UPDATE messages and advertised to its peers.

In summary, the Adj-RIBs-In contain unprocessed routing information that has been advertised to the local BGP speaker by its peers; the Loc-RIB contains the routes that have been selected by the local BGP speaker's Decision Process; and the Adj-RIBs-Out organize the routes for advertisement to specific peers by means of the local speaker's UPDATE messages.

Although the conceptual model distinguishes between Adj-RIBs-In, Loc-RIB, and Adj-RIBs-Out, this neither implies nor requires that an implementation must maintain three separate copies of the routing information. The choice of implementation (for example, 3 copies of the information vs 1 copy with pointers) is not constrained by the protocol.

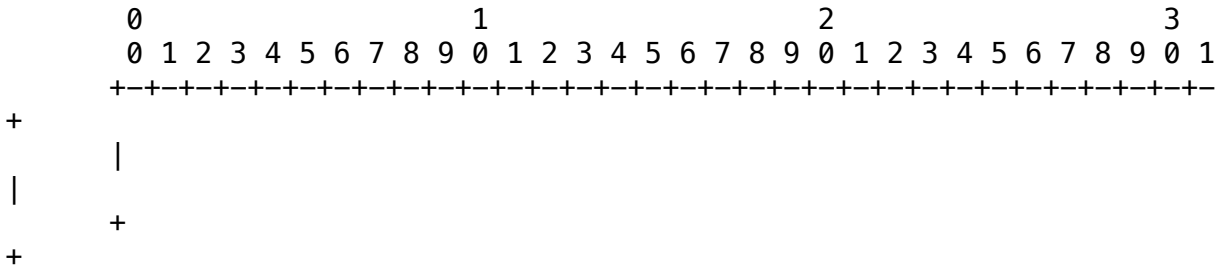
4. Message Formats

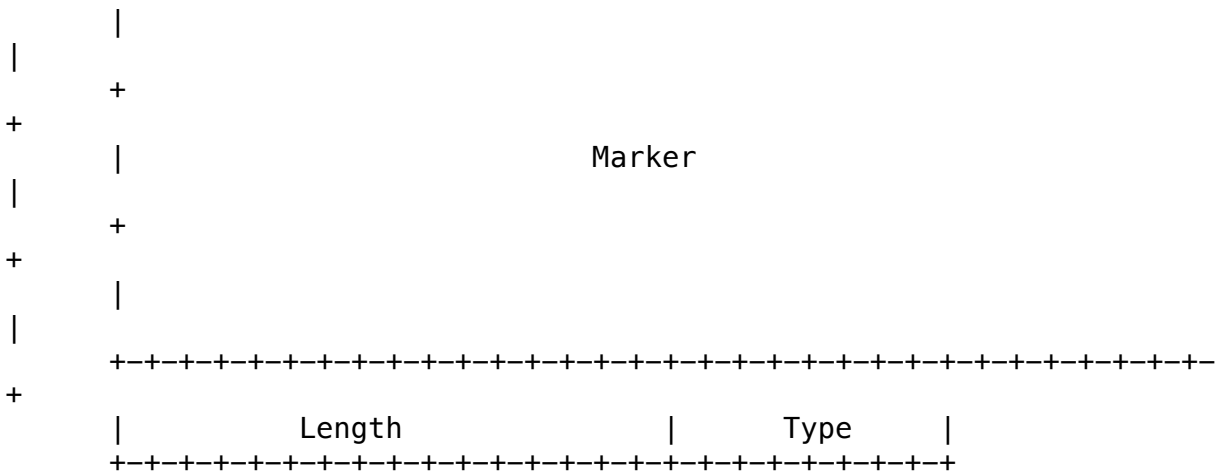
This section describes message formats used by BGP.

Messages are sent over a reliable transport protocol connection. A message is processed only after it is entirely received. The maximum message size is 4096 octets. All implementations are required to support this maximum message size. The smallest message that may be sent consists of a BGP header without a data portion, or 19 octets.

4.1 Message Header Format

Each message has a fixed-size header. There may or may not be a data portion following the header, depending on the message type. The layout of these fields is shown below:





Marker:

This 16-octet field contains a value that the receiver of the message can predict. If the Type of the message is OPEN, or if the OPEN message carries no Authentication Information (as an Optional Parameter), then the Marker must be all ones. Otherwise, the value of the marker can be predicted by some a computation specified as part of the authentication mechanism (which is specified as part of the Authentication Information) used. The Marker can be used to detect loss of synchronization between a pair of BGP peers, and to authenticate incoming BGP messages.

Length:

This 2-octet unsigned integer indicates the total length of the message, including the header, in octets. Thus, e.g., it allows one to locate in the transport-level stream the (Marker field of the) next message. The value of the Length field must

always be at least 19 and no greater than 4096, and may be further constrained, depending on the message type. No "padding" of extra data after the message is allowed, so the Length field must have the smallest value required given the rest of the message.

Type:

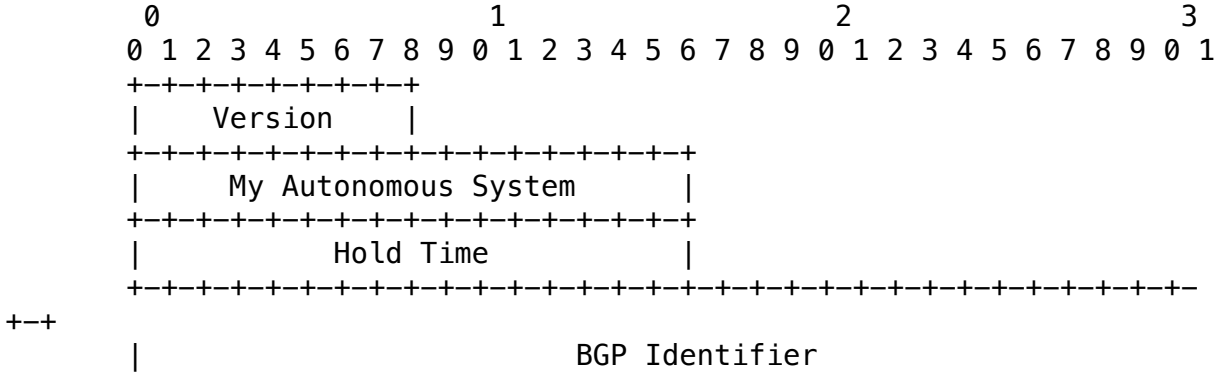
This 1-octet unsigned integer indicates the type code of the message. The following type codes are defined:

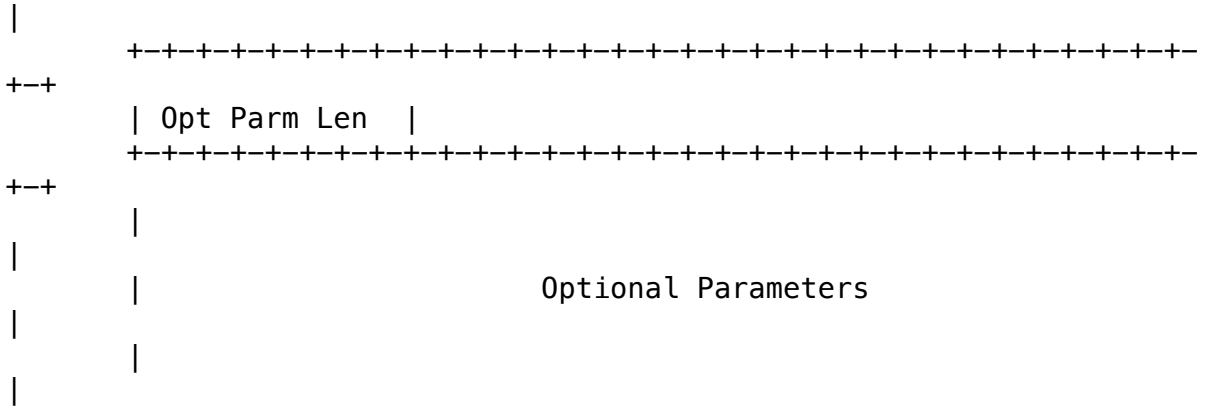
1 - OPEN
2 - UPDATE
3 - NOTIFICATION
4 - KEEPALIVE

4.2 OPEN Message Format

After a transport protocol connection is established, the first message sent by each side is an OPEN message. If the OPEN message is acceptable, a KEEPALIVE message confirming the OPEN is sent back. Once the OPEN is confirmed, UPDATE, KEEPALIVE, and NOTIFICATION messages may be exchanged.

In addition to the fixed-size BGP header, the OPEN message contains the following fields:





Version:

This 1-octet unsigned integer indicates the protocol version number of the message. The current BGP version number is 4.

My Autonomous System:

This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Hold Time:

This 2-octet unsigned integer indicates the number of seconds that the sender proposes for the value of the Hold Timer. Upon receipt of an OPEN message, a BGP speaker MUST calculate the value of the Hold Timer by using the smaller of its configured Hold Time and the Hold Time received in the OPEN message.

The
Hold Time MUST be either zero or at least three seconds. An implementation may reject connections on the basis of the Hold Time. The calculated value indicates the maximum number of seconds that may elapse between the receipt of successive KEEPALIVE, and/or UPDATE messages by the sender.

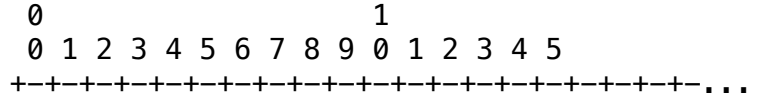
BGP Identifier:
This 4-octet unsigned integer indicates the BGP Identifier of the sender. A given BGP speaker sets the value of its BGP Identifier to an IP address assigned to that BGP speaker. The value of the BGP Identifier is determined on startup and is the same for every local interface and every BGP peer.

Optional Parameters Length:

This 1-octet unsigned integer indicates the total length of the Optional Parameters field in octets. If the value of this field is zero, no Optional Parameters are present.

Optional Parameters:

This field may contain a list of optional parameters, where each parameter is encoded as a <Parameter Type, Parameter Length, Parameter Value> triplet.



```
                |  Parm. Type   | Parm. Length  |  Parameter Value
(variable)
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...
```

Parameter Type is a one octet field that unambiguously identifies individual parameters. Parameter Length is a one octet field that contains the length of the Parameter Value field in octets. Parameter Value is a variable length field that is interpreted according to the value of the Parameter Type field.

This document defines the following Optional Parameters:

a) Authentication Information (Parameter Type 1):

This optional parameter may be used to authenticate a BGP peer. The Parameter Value field contains a 1-octet Authentication Code followed by a variable length Authentication Data.

```
                0 1 2 3 4 5 6 7 8
                +-+-+-+-+-+-+-+-+
                |  Auth. Code   |
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+
                |
|
                |              Authentication Data
|
                |
|
                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+
```

Authentication Code:

This 1-octet unsigned integer indicates the authentication mechanism being used. Whenever an authentication mechanism is specified for use within BGP, three things must be included in the specification:
- the value of the Authentication Code which indicates use of the mechanism,
- the form and meaning of the Authentication Data,

and
- the algorithm for computing values of Marker fields.

Note that a separate authentication mechanism may be used in establishing the transport level connection.

Authentication Data:

The form and meaning of this field is a variable-length field depend on the Authentication Code.

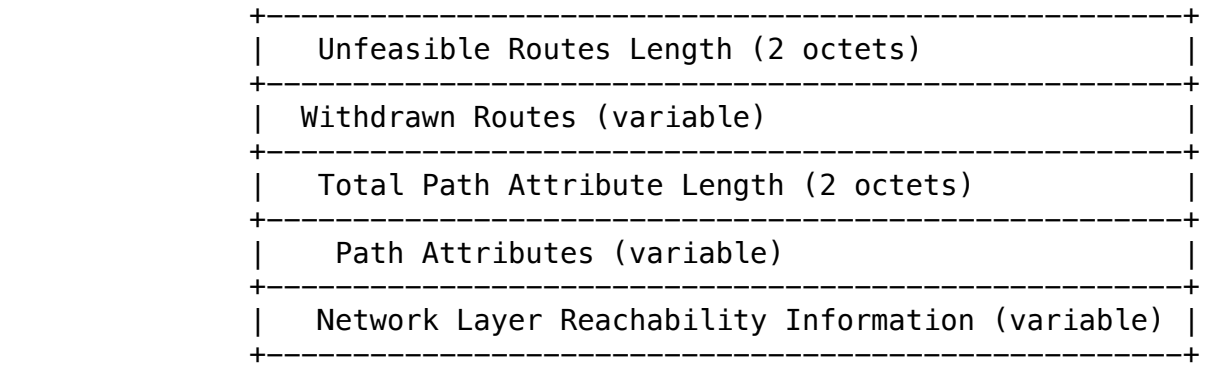
The minimum length of the OPEN message is 29 octets (including message header).

4.3 UPDATE Message Format

UPDATE messages are used to transfer routing information between BGP peers. The information in the UPDATE packet can be used to construct a graph describing the relationships of the various Autonomous Systems. By applying rules to be discussed, routing information loops and some other anomalies may be detected and removed from inter-AS routing.

An UPDATE message is used to advertise a single feasible route to a peer, or to withdraw multiple unfeasible routes from service (see 3.1). An UPDATE message may simultaneously advertise a feasible route and withdraw multiple unfeasible routes from service. The UPDATE message always includes the fixed-size BGP header, and can optionally

include the other fields as shown below:



Unfeasible Routes Length:

This 2-octets unsigned integer indicates the total length of the Withdrawn Routes field in octets. Its value must allow the

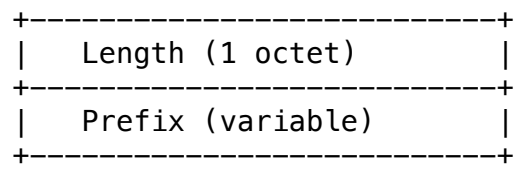
length of the Network Layer Reachability Information field to be determined as specified below.

A value of 0 indicates that no routes are being withdrawn from service, and that the WITHDRAWN ROUTES field is not present in this UPDATE message.

Withdrawn Routes:

This is a variable length field that contains a list of IP address prefixes for the routes that are being withdrawn from

service. Each IP address prefix is encoded as a 2-tuple of the
form <length, prefix>, whose fields are described below:



The use and the meaning of these fields are as follows:

a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains an IP address prefix followed by enough trailing bits to make the end of the field fall on an
octet boundary. Note that the value of trailing bits is irrelevant.

Total Path Attribute Length:

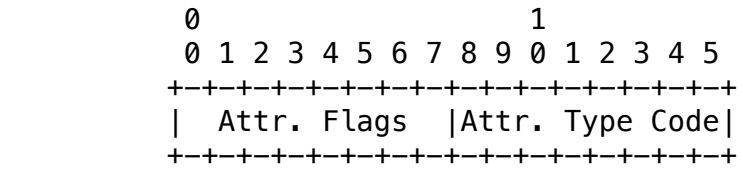
This 2-octet unsigned integer indicates the total length of the
Path Attributes field in octets. Its value must allow the length of the Network Layer Reachability field to be determined
as specified below.

A value of 0 indicates that no Network Layer Reachability Information field is present in this UPDATE message.

Path Attributes:

A variable length sequence of path attributes is present in every UPDATE. Each path attribute is a triple <attribute type, attribute length, attribute value> of variable length.

Attribute Type is a two-octet field that consists of the Attribute Flags octet followed by the Attribute Type Code octet.



The high-order bit (bit 0) of the Attribute Flags octet is the Optional bit. It defines whether the attribute is optional (if set to 1) or well-known (if set to 0).

The second high-order bit (bit 1) of the Attribute Flags octet is the Transitive bit. It defines whether an optional attribute is transitive (if set to 1) or non-transitive (if set to 0). For well-known attributes, the Transitive bit must be set to 1. (See Section 5 for a discussion of transitive attributes.)

The third high-order bit (bit 2) of the Attribute Flags octet is the Partial bit. It defines whether the information contained in the optional transitive attribute is partial (if set to 1) or complete (if set to 0). For well-known attributes and for optional non-transitive attributes the Partial bit must be set to 0.

The fourth high-order bit (bit 3) of the Attribute Flags

octet
is the Extended Length bit. It defines whether the Attribute
Length is one octet (if set to 0) or two octets (if set to
1).
Extended Length may be used only if the length of the
attribute
value is greater than 255 octets.

The lower-order four bits of the Attribute Flags octet are .
unused. They must be zero (and must be ignored when
received).

The Attribute Type Code octet contains the Attribute Type
Code.

Currently defined Attribute Type Codes are discussed in
Section
5.

If the Extended Length bit of the Attribute Flags octet is
set
to 0, the third octet of the Path Attribute contains the
length
of the attribute data in octets.

If the Extended Length bit of the Attribute Flags octet is
set
to 1, then the third and the fourth octets of the path
attribute contain the length of the attribute data in octets.

The remaining octets of the Path Attribute represent the
attribute value and are interpreted according to the
Attribute
Flags and the Attribute Type Code. The supported Attribute
Type
Codes, their attribute values and uses are the following:

a) ORIGIN (Type Code 1):

ORIGIN is a well-known mandatory attribute that defines the origin of the path information. The data octet can assume the following values:

Value	Meaning
0	IGP - Network Layer Reachability Information is interior to the originating AS
1	EGP - Network Layer Reachability Information learned via EGP
2	INCOMPLETE - Network Layer Reachability Information learned by some other means

Its usage is defined in 5.1.1

b) AS_PATH (Type Code 2):

AS_PATH is a well-known mandatory attribute that is composed of a sequence of AS path segments. Each AS path segment is represented by a triple <path segment type, path segment length, path segment value>.

The path segment type is a 1-octet long field with the following values defined:

Value	Segment Type
1	AS_SET: unordered set of ASs a route in the

UPDATE message has traversed

2 AS_SEQUENCE: ordered set of ASs a route in the UPDATE message has traversed

The path segment length is a 1-octet long field containing the number of ASs in the path segment value field.

The path segment value field contains one or more AS numbers, each encoded as a 2-octets long field.

Usage of this attribute is defined in 5.1.2.

c) NEXT_HOP (Type Code 3):

This is a well-known mandatory attribute that defines the IP address of the border router that should be used as the next hop to the destinations listed in the Network Layer Reachability field of the UPDATE message.

Usage of this attribute is defined in 5.1.3.

d) MULTI_EXIT_DISC (Type Code 4):

This is an optional non-transitive attribute that is a four octet non-negative integer. The value of this attribute may be used by a BGP speaker's decision process to discriminate among multiple exit points to a neighboring autonomous system.

Its usage is defined in 5.1.4.

e) LOCAL_PREF (Type Code 5):

LOCAL_PREF is a well-known mandatory attribute that is a four octet non-negative integer. It is used by a BGP speaker to inform other internal peers of the advertising speaker's degree of preference for an advertised route. Usage of this attribute is described in 5.1.5.

f) ATOMIC_AGGREGATE (Type Code 6)

ATOMIC_AGGREGATE is a well-known discretionary attribute of length 0. It is used by a BGP speaker to inform other BGP speakers that the local system selected a less specific route without selecting a more specific route which is included in it. Usage of this attribute is described in

5.1.6.

g) AGGREGATOR (Type Code 7)

AGGREGATOR is an optional transitive attribute of length 6.
The attribute contains the last AS number that formed the aggregate route (encoded as 2 octets), followed by the IP address of the BGP speaker that formed the aggregate route (encoded as 4 octets). Usage of this attribute is described in 5.1.7

Network Layer Reachability Information:

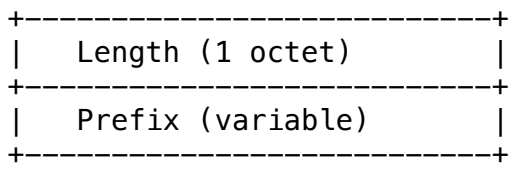
This variable length field contains a list of IP address prefixes. The length in octets of the Network Layer Reachability Information is not encoded explicitly, but can be calculated as:

UPDATE message Length - 23 - Total Path Attributes Length - Unfeasible Routes Length

where UPDATE message Length is the value encoded in the fixed-

size BGP header, Total Path Attribute Length and Unfeasible Routes Length are the values encoded in the variable part of the UPDATE message, and 23 is a combined length of the fixed-size BGP header, the Total Path Attribute Length field and the Unfeasible Routes Length field.

Reachability information is encoded as one or more 2-tuples of the form <length, prefix>, whose fields are described below:



The use and the meaning of these fields are as follows:

a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains IP address prefixes followed by enough trailing bits to make the end of the field fall on an octet boundary. Note that the value of the trailing bits is irrelevant.

The minimum length of the UPDATE message is 23 octets -- 19 octets for the fixed header + 2 octets for the Unfeasible Routes Length + 2 octets for the Total Path Attribute Length (the value of Unfeasible Routes Length is 0 and the value of Total Path Attribute Length is 0).

An UPDATE message can advertise at most one route, which may be described by several path attributes. All path attributes contained in a given UPDATE messages apply to the destinations carried in the Network Layer Reachability Information field of the UPDATE message.

An UPDATE message can list multiple routes to be withdrawn from service. Each such route is identified by its destination (expressed as an IP prefix), which unambiguously identifies the route in the context of the BGP speaker - BGP speaker connection to which it has been previously been advertised.

An UPDATE message may advertise only routes to be withdrawn from service, in which case it will not include path attributes or Network Layer Reachability Information. Conversely, it may advertise only a feasible route, in which case the WITHDRAWN ROUTES field need not be present.

4.4 KEEPALIVE Message Format

BGP does not use any transport protocol-based keep-alive mechanism to determine if peers are reachable. Instead, KEEPALIVE messages are exchanged between peers often enough as not to cause the Hold Timer to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval. KEEPALIVE messages MUST NOT be sent more frequently than one per second. An implementation MAY adjust the rate at which it sends KEEPALIVE messages as a function of the Hold Time interval.

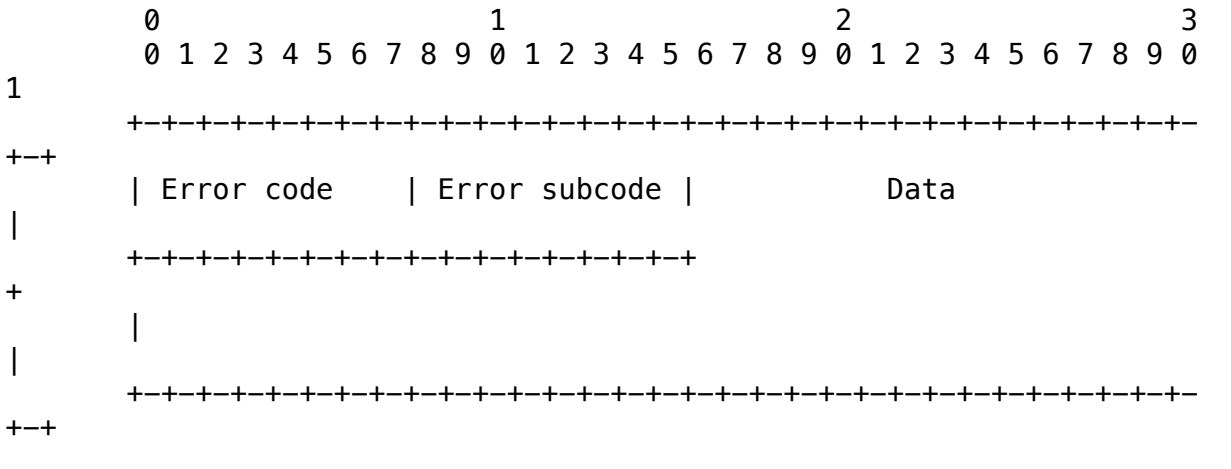
If the negotiated Hold Time interval is zero, then periodic KEEPALIVE messages MUST NOT be sent.

KEEPALIVE message consists of only message header and has a length of 19 octets.

4.5 NOTIFICATION Message Format

A NOTIFICATION message is sent when an error condition is detected. The BGP connection is closed immediately after sending it.

In addition to the fixed-size BGP header, the NOTIFICATION message contains the following fields:



Error Code:

This 1-octet unsigned integer indicates the type of NOTIFICATION. The following Error Codes have been defined:

Error Code	Symbolic Name	Reference
1	Message Header Error	Section 6.1
2	OPEN Message Error	Section 6.2

3	UPDATE Message Error	Section 6.3
4	Hold Timer Expired	Section 6.5
5	Finite State Machine Error	Section 6.6
6	Cease	Section 6.7

Error subcode:

This 1-octet unsigned integer provides more specific information about the nature of the reported error. Each Error Code may have one or more Error Subcodes associated with it. If no appropriate Error Subcode is defined, then a zero (Unspecific) value is used for the Error Subcode field.

Message Header Error subcodes:

1 - Connection Not Synchronized.
2 - Bad Message Length.
3 - Bad Message Type.

OPEN Message Error subcodes:

1 - Unsupported Version Number.
2 - Bad Peer AS.
3 - Bad BGP Identifier.
4 - Unsupported Optional Parameter.
5 - Authentication Failure.
6 - Unacceptable Hold Time.

UPDATE Message Error subcodes:

1 - Malformed Attribute List.

2 - Unrecognized Well-known Attribute.
3 - Missing Well-known Attribute.
4 - Attribute Flags Error.
5 - Attribute Length Error.
6 - Invalid ORIGIN Attribute
8 - Invalid NEXT_HOP Attribute.
9 - Optional Attribute Error.
10 - Invalid Network Field.
11 - Malformed AS_PATH.

Data:

This variable-length field is used to diagnose the reason for the NOTIFICATION. The contents of the Data field depend upon the Error Code and Error Subcode. See Section 6 below for more details.

Note that the length of the Data field can be determined from the message Length field by the formula:

Message Length = 21 + Data Length

The minimum length of the NOTIFICATION message is 21 octets (including message header).

5. Path Attributes

This section discusses the path attributes of the UPDATE message.

Path attributes fall into four separate categories:

1. Well-known mandatory.
2. Well-known discretionary.
3. Optional transitive.
4. Optional non-transitive.

Well-known attributes must be recognized by all BGP implementations.
Some of these attributes are mandatory and must be included in every
UPDATE message that contains NLRI. Others are discretionary and may
or may not be sent in a particular UPDATE message.

All well-known attributes must be passed along (after proper updating, if necessary) to other BGP peers.

In addition to well-known attributes, each path may contain one or more optional attributes. It is not required or expected that all BGP implementations support all optional attributes. The handling of
an unrecognized optional attribute is determined by the setting of the Transitive bit in the attribute flags octet. Paths with unrecognized transitive optional attributes should be accepted. If a
path with unrecognized transitive optional attribute is accepted and
passed along to other BGP peers, then the unrecognized transitive optional attribute of that path must be passed along with the path to
other BGP peers with the Partial bit in the Attribute Flags octet set
to 1. If a path with recognized transitive optional attribute is accepted and passed along to other BGP peers and the Partial bit in the Attribute Flags octet is set to 1 by some previous AS, it is not
set back to 0 by the current AS. Unrecognized non-transitive optional
attributes must be quietly ignored and not passed along to other BGP
peers.

New transitive optional attributes may be attached to the path by the
originator or by any other AS in the path. If they are not attached
by the originator, the Partial bit in the Attribute Flags octet is set to 1. The rules for attaching new non-transitive optional attributes will depend on the nature of the specific attribute.

The
documentation of each new non-transitive optional attribute will be expected to include such rules. (The description of the MULTI_EXIT_DISC attribute gives an example.) All optional attributes
(both transitive and non-transitive) may be updated (if appropriate)
by ASs in the path.

The sender of an UPDATE message should order path attributes within the UPDATE message in ascending order of attribute type. The receiver of an UPDATE message must be prepared to handle path attributes within the UPDATE message that are out of order.

The same attribute cannot appear more than once within the Path Attributes field of a particular UPDATE message.

The mandatory category refers to an attribute which must be present in both IBGP and EBGP exchanges if NLRI are contained in the UPDATE message. Attributes classified as optional for the purpose of the protocol extension mechanism may be purely discretionary, or discretionary, required, or disallowed in certain contexts.

attribute	EBGP	IBGP
ORIGIN	mandatory	mandatory
AS_PATH	mandatory	mandatory
NEXT_HOP	mandatory	mandatory
MULTI_EXIT_DISC	discretionary	discretionary
LOCAL_PREF	disallowed	required
ATOMIC_AGGREGATE	see section 5.1.6 and 9.1.4	
AGGREGATOR	discretionary	discretionary

5.1 Path Attribute Usage

The usage of each BGP path attributes is described in the following clauses.

5.1.1 ORIGIN

ORIGIN is a well-known mandatory attribute. The ORIGIN attribute shall be generated by the autonomous system that originates the associated routing information. It shall be included in the UPDATE messages of all BGP speakers that choose to propagate this information to other BGP speakers.

5.1.2 AS_PATH

AS_PATH is a well-known mandatory attribute. This attribute

identifies the autonomous systems through which routing information carried in this UPDATE message has passed. The components of this list can be AS_SETs or AS_SEQUENCEs.

When a BGP speaker propagates a route which it has learned from another BGP speaker's UPDATE message, it shall modify the route's AS_PATH attribute based on the location of the BGP speaker to which the route will be sent:

a) When a given BGP speaker advertises the route to an internal peer, the advertising speaker shall not modify the AS_PATH attribute associated with the route.

b) When a given BGP speaker advertises the route to an external peer, then the advertising speaker shall update the AS_PATH

attribute as follows:

1) if the first path segment of the AS_PATH is of type AS_SEQUENCE, the local system shall prepend its own AS number as the last element of the sequence (put it in the leftmost position)

2) if the first path segment of the AS_PATH is of type AS_SET, the local system shall prepend a new path segment of type AS_SEQUENCE to the AS_PATH, including its own AS number in that segment.

When a BGP speaker originates a route then:

a) the originating speaker shall include its own AS number in the AS_PATH attribute of all UPDATE messages sent to an external peer. (In this case, the AS number of the originating speaker's autonomous system will be the only entry in the AS_PATH attribute).

b) the originating speaker shall include an empty AS_PATH attribute in all UPDATE messages sent to internal peers. (An empty AS_PATH attribute is one whose length field contains the value zero).

5.1.3 NEXT_HOP

The NEXT_HOP path attribute defines the IP address of the border router that should be used as the next hop to the destinations listed in the UPDATE message. When advertising a NEXT_HOP attribute to an

external peer, a router may use one of its own interface addresses in the NEXT_HOP attribute provided the external peer to which the route is being advertised shares a common subnet with the NEXT_HOP address.

This is known as a "first party" NEXT_HOP attribute. A BGP speaker can advertise to an external peer an interface of any internal peer router in the NEXT_HOP attribute provided the external peer to which the route is being advertised shares a common subnet with the NEXT_HOP address. This is known as a "third party" NEXT_HOP attribute. A BGP speaker can advertise any adjacent router in the NEXT_HOP attribute provided that the IP address of this router was learned from an external peer and the external peer to which the route is being advertised shares a common subnet with the NEXT_HOP address. This is a second form of "third party" NEXT_HOP attribute.

Normally the NEXT_HOP attribute is chosen such that the shortest available path will be taken. A BGP speaker must be able to support disabling advertisement of third party NEXT_HOP attributes to handle imperfectly bridged media.

A BGP speaker must never advertise an address of a peer to that peer as a NEXT_HOP, for a route that the speaker is originating. A BGP speaker must never install a route with itself as the next hop.

When a BGP speaker advertises the route to an internal peer, the advertising speaker should not modify the NEXT_HOP attribute associated with the route. When a BGP speaker receives the route via an internal link, it may forward packets to the NEXT_HOP address if the address contained in the attribute is on a common subnet with the local and remote BGP speakers.

5.1.4 MULTI_EXIT_DISC

The MULTI_EXIT_DISC attribute may be used on external (inter-AS) links to discriminate among multiple exit or entry points to the same neighboring AS. The value of the MULTI_EXIT_DISC attribute is a four

octet unsigned number which is called a metric. All other factors being equal, the exit or entry point with lower metric should be preferred. If received over external links, the MULTI_EXIT_DISC attribute MAY be propagated over internal links to other BGP speakers within the same AS. The MULTI_EXIT_DISC attribute received from a neighboring AS MUST NOT be propagated to other neighboring ASs.

A BGP speaker MUST IMPLEMENT a mechanism based on local configuration which allows the MULTI_EXIT_DISC attribute to be removed from a route. This MAY be done either prior to or after determining the degree of preference of the route and performing route selection

(decision process phases 1 and 2).

An implementation MAY also (based on local configuration) alter the value of the MULTI_EXIT_DISC attribute received over an external link. If it does so, it shall do so prior to determining the degree of preference of the route and performing route selection (decision process phases 1 and 2).

5.1.5 LOCAL_PREF

LOCAL_PREF is a well-known mandatory attribute that SHALL be included in all UPDATE messages that a given BGP speaker sends to the other internal peers. A BGP speaker SHALL calculate the degree of preference for each external route and include the degree of preference when advertising a route to its internal peers. The higher degree of preference MUST be preferred. A BGP speaker shall use the degree of preference learned via LOCAL_PREF in its decision process (see section 9.1.1).

A BGP speaker MUST NOT include this attribute in UPDATE messages that it sends to external peers. If it is contained in an UPDATE message that is received from an external peer, then this attribute MUST be ignored by the receiving speaker.

5.1.6 ATOMIC_AGGREGATE

ATOMIC_AGGREGATE is a well-known discretionary attribute. If a BGP speaker, when presented with a set of overlapping routes from one of its peers (see 9.1.4), selects the less specific route without selecting the more specific one, then the local system MUST attach the ATOMIC_AGGREGATE attribute to the route when propagating it to other BGP speakers (if that attribute is not already present in the received less specific route). A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute MUST NOT remove the attribute from the route when propagating it to other speakers. A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute MUST NOT make any NLRI of that route more specific (as defined in 9.1.4) when advertising this route to other BGP speakers. A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute needs to be cognizant of the fact that the actual path to destinations, as specified in the NLRI of the route, while having the loop-free property, may traverse ASs that are not listed in the AS_PATH attribute.

5.1.7 AGGREGATOR

AGGREGATOR is an optional transitive attribute which may be

included
in updates which are formed by aggregation (see Section 9.2.4.2).
A
BGP speaker which performs route aggregation may add the AGGREGATOR
attribute which shall contain its own AS number and IP address.

6. BGP Error Handling.

This section describes actions to be taken when errors are detected
while processing BGP messages.

When any of the conditions described here are detected, a
NOTIFICATION message with the indicated Error Code, Error Subcode,
and Data fields is sent, and the BGP connection is closed. If no
Error Subcode is specified, then a zero must be used.

The phrase "the BGP connection is closed" means that the transport
protocol connection has been closed and that all resources for that
BGP connection have been deallocated. Routing table entries
associated with the remote peer are marked as invalid. The fact
that
the routes have become invalid is passed to other BGP peers before
the routes are deleted from the system.

Unless specified explicitly, the Data field of the NOTIFICATION
message that is sent to indicate an error is empty.

6.1 Message Header error handling.

All errors detected while processing the Message Header are
indicated
by sending the NOTIFICATION message with Error Code Message Header
Error. The Error Subcode elaborates on the specific nature of the
error.

The expected value of the Marker field of the message header is all
ones if the message type is OPEN. The expected value of the Marker
field for all other types of BGP messages determined based on the
presence of the Authentication Information Optional Parameter in
the
BGP OPEN message and the actual authentication mechanism (if the
Authentication Information in the BGP OPEN message is present). If
the Marker field of the message header is not the expected one,
then
a synchronization error has occurred and the Error Subcode is set
to

Connection Not Synchronized.

If the Length field of the message header is less than 19 or greater than 4096, or if the Length field of an OPEN message is less than the minimum length of the OPEN message, or if the Length field of an UPDATE message is less than the minimum length of the UPDATE message, or if the Length field of a KEEPALIVE message is not equal to 19, or if the Length field of a NOTIFICATION message is less than the minimum length of the NOTIFICATION message, then the Error Subcode is set to Bad Message Length. The Data field contains the erroneous Length field.

If the Type field of the message header is not recognized, then the Error Subcode is set to Bad Message Type. The Data field contains the erroneous Type field.

6.2 OPEN message error handling.

All errors detected while processing the OPEN message are indicated by sending the NOTIFICATION message with Error Code OPEN Message Error. The Error Subcode elaborates on the specific nature of the error.

If the version number contained in the Version field of the received OPEN message is not supported, then the Error Subcode is set to Unsupported Version Number. The Data field is a 1-octet unsigned integer, which indicates the largest locally supported version number less than the version the remote BGP peer bid (as indicated in the

received OPEN message).

If the Autonomous System field of the OPEN message is unacceptable, then the Error Subcode is set to Bad Peer AS. The determination of acceptable Autonomous System numbers is outside the scope of this protocol.

If the Hold Time field of the OPEN message is unacceptable, then the Error Subcode MUST be set to Unacceptable Hold Time. An implementation MUST reject Hold Time values of one or two seconds. An implementation MAY reject any proposed Hold Time. An implementation which accepts a Hold Time MUST use the negotiated value for the Hold Time.

If the BGP Identifier field of the OPEN message is syntactically incorrect, then the Error Subcode is set to Bad BGP Identifier. Syntactic correctness means that the BGP Identifier field represents a valid IP host address.

If one of the Optional Parameters in the OPEN message is not

recognized, then the Error Subcode is set to Unsupported Optional Parameters.

If the OPEN message carries Authentication Information (as an Optional Parameter), then the corresponding authentication procedure is invoked. If the authentication procedure (based on Authentication Code and Authentication Data) fails, then the Error Subcode is set to Authentication Failure.

6.3 UPDATE message error handling.

All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

Error checking of an UPDATE message begins by examining the path attributes. If the Unfeasible Routes Length or Total Attribute Length is too large (i.e., if Unfeasible Routes Length + Total Attribute Length + 23 exceeds the message Length), then the Error Subcode is set to Malformed Attribute List.

If any recognized attribute has Attribute Flags that conflict with the Attribute Type Code, then the Error Subcode is set to Attribute Flags Error. The Data field contains the erroneous attribute (type, length and value).

If any recognized attribute has Attribute Length that conflicts with the expected length (based on the attribute type code), then the Error Subcode is set to Attribute Length Error. The Data field contains the erroneous attribute (type, length and value).

If any of the mandatory well-known attributes are not present, then the Error Subcode is set to Missing Well-known Attribute. The Data field contains the Attribute Type Code of the missing well-known attribute.

If any of the mandatory well-known attributes are not recognized, then the Error Subcode is set to Unrecognized Well-known Attribute. The Data field contains the unrecognized attribute (type, length and value).

If the ORIGIN attribute has an undefined value, then the Error Subcode is set to Invalid Origin Attribute. The Data field contains

the unrecognized attribute (type, length and value).

If the NEXT_HOP attribute field is syntactically incorrect, then the Error Subcode is set to Invalid NEXT_HOP Attribute. The Data field contains the incorrect attribute (type, length and value). Syntactic correctness means that the NEXT_HOP attribute represents a valid IP host address. Semantic correctness applies only to the external BGP links. It means that the interface associated with the IP address, as specified in the NEXT_HOP attribute, shares a common subnet with the receiving BGP speaker and is not the IP address of the receiving BGP speaker. If the NEXT_HOP attribute is semantically incorrect, the error should be logged, and the the route should be ignored. In this case, no NOTIFICATION message should be sent.

The AS_PATH attribute is checked for syntactic correctness. If the path is syntactically incorrect, then the Error Subcode is set to Malformed AS_PATH.

The information carried by the AS_PATH attribute is checked for AS loops. AS loop detection is done by scanning the full AS path (as specified in the AS_PATH attribute), and checking that the autonomous system number of the local system does not appear in the AS path. If the autonomous system number appears in the AS path the route may be stored in the Adj-RIB-In, but unless the router is configured to accept routes with its own autonomous system in the AS path, the route shall not be passed to the BGP Decision Process. Operations of a router that is configured to accept routes with its own autonomous system number in the AS path are outside the scope of this document.

If an optional attribute is recognized, then the value of this

attribute is checked. If an error is detected, the attribute is discarded, and the Error Subcode is set to Optional Attribute Error.
The Data field contains the attribute (type, length and value).

If any attribute appears more than once in the UPDATE message, then the Error Subcode is set to Malformed Attribute List.

The NLRI field in the UPDATE message is checked for syntactic validity. If the field is syntactically incorrect, then the Error Subcode is set to Invalid Network Field.

An UPDATE message that contains correct path attributes, but no NLRI,
shall be treated as a valid UPDATE message.

6.4 NOTIFICATION message error handling.

If a peer sends a NOTIFICATION message, and there is an error in that
message, there is unfortunately no means of reporting this error via
a subsequent NOTIFICATION message. Any such error, such as an unrecognized Error Code or Error Subcode, should be noticed, logged locally, and brought to the attention of the administration of the peer. The means to do this, however, lies outside the scope of this
document.

6.5 Hold Timer Expired error handling.

If a system does not receive successive KEEPALIVE and/or UPDATE and/or NOTIFICATION messages within the period specified in the Hold Time field of the OPEN message, then the NOTIFICATION message with Hold Timer Expired Error Code must be sent and the BGP connection closed.

6.6 Finite State Machine error handling.

Any error detected by the BGP Finite State Machine (e.g., receipt of an unexpected event) is indicated by sending the NOTIFICATION message with Error Code Finite State Machine Error.

6.7 Cease.

In absence of any fatal errors (that are indicated in this section), a BGP peer may choose at any given time to close its BGP connection by sending the NOTIFICATION message with Error Code Cease. However, the Cease NOTIFICATION message must not be used when a fatal error indicated by this section does exist.

6.8 Connection collision detection.

If a pair of BGP speakers try simultaneously to establish a TCP connection to each other, then two parallel connections between this pair of speakers might well be formed. We refer to this situation as connection collision. Clearly, one of these connections must be closed.

Based on the value of the BGP Identifier a convention is established for detecting which BGP connection is to be preserved when a collision does occur. The convention is to compare the BGP Identifiers of the peers involved in the collision and to retain only the connection initiated by the BGP speaker with the higher-valued BGP Identifier.

Upon receipt of an OPEN message, the local system must examine all of its connections that are in the OpenConfirm state. A BGP speaker may also examine connections in an OpenSent state if it knows the BGP Identifier of the peer by means outside of the protocol. If among these connections there is a connection to a remote BGP speaker whose BGP Identifier equals the one in the OPEN message, then the local system performs the following collision resolution procedure:

1. The BGP Identifier of the local system is compared to the BGP Identifier of the remote system (as specified in the OPEN message).

2. If the value of the local BGP Identifier is less than the remote one, the local system closes BGP connection that already exists (the one that is already in the OpenConfirm state), and accepts BGP connection initiated by the remote system.

3. Otherwise, the local system closes newly created BGP connection (the one associated with the newly received OPEN message), and continues to use the existing one (the one that is already in the OpenConfirm state).

Comparing BGP Identifiers is done by treating them as (4-octet long) unsigned integers.

Unless allowed via configuration, a connection collision with an existing BGP connection that is in Established state causes closing of the newly created connection.

Note that a connection collision cannot be detected with

connections that are in Idle, or Connect, or Active states.

Closing the BGP connection (that results from the collision resolution procedure) is accomplished by sending the NOTIFICATION message with the Error Code Cease.

7. BGP Version Negotiation.

BGP speakers may negotiate the version of the protocol by making multiple attempts to open a BGP connection, starting with the highest version number each supports. If an open attempt fails with an Error Code OPEN Message Error, and an Error Subcode Unsupported Version Number, then the BGP speaker has available the version number it tried, the version number its peer tried, the version number passed by its peer in the NOTIFICATION message, and the version numbers that it supports. If the two peers do support one or more common versions, then this will allow them to rapidly determine the highest common version. In order to support BGP version negotiation, future versions of BGP must retain the format of the OPEN and NOTIFICATION messages.

8. BGP Finite State machine.

This section specifies BGP operation in terms of a Finite State

Machine (FSM). Following is a brief summary and overview of BGP operations by state as determined by this FSM. A condensed version of the BGP FSM is found in Appendix 1.

Initially BGP is in the Idle state.

Idle state:

In this state BGP refuses all incoming BGP connections. No resources are allocated to the peer. In response to the Start event (initiated by either system or operator) the local system initializes all BGP resources, starts the ConnectRetry timer, initiates a transport connection to other BGP peer, while listening for connection that may be initiated by the remote BGP peer, and changes its state to Connect. The exact value of the ConnectRetry timer is a local matter, but should be sufficiently large to allow TCP initialization.

If a BGP speaker detects an error, it shuts down the connection and changes its state to Idle. Getting out of the Idle state requires generation of the Start event. If such an event is generated automatically, then persistent BGP errors may result in persistent flapping of the speaker. To avoid such a condition it is recommended that Start events should not be generated immediately for a peer that was previously transitioned to Idle due to an error. For a peer that was previously transitioned to Idle due to an error, the time

between consecutive generation of Start events, if such events are generated automatically, shall exponentially increase. The

value of the initial timer shall be 60 seconds. The time shall
be doubled for each consecutive retry.

Any other event received in the Idle state is ignored.

Connect state:

In this state BGP is waiting for the transport protocol connection to be completed.

If the transport protocol connection succeeds, the local system
clears the ConnectRetry timer, completes initialization, sends
an OPEN message to its peer, and changes its state to OpenSent.

If the transport protocol connect fails (e.g., retransmission timeout), the local system restarts the ConnectRetry timer, continues to listen for a connection that may be initiated by the remote BGP peer, and changes its state to Active state.

In response to the ConnectRetry timer expired event, the local
system restarts the ConnectRetry timer, initiates a transport connection to other BGP peer, continues to listen for a connection that may be initiated by the remote BGP peer, and stays in the Connect state.

Start event is ignored in the Connect state.

In response to any other event (initiated by either system or operator), the local system releases all BGP resources associated with this connection and changes its state to Idle.

Active state:

In this state BGP is trying to acquire a peer by listening for
and accepting a transport protocol connection.

If the transport protocol connection succeeds, the local system
clears the ConnectRetry timer, completes initialization, sends
an OPEN message to its peer, sets its Hold Timer to a large value, and changes its state to OpenSent. A Hold Timer value of 4 minutes is suggested.

In response to the ConnectRetry timer expired event, the local system restarts the ConnectRetry timer, initiates a transport connection to other BGP peer, continues to listen for a connection that may be initiated by the remote BGP peer, and

changes its state to Connect.

If the local system allows BGP connections with unconfigured peers, then when the local system detects that a remote peer is trying to establish a BGP connection to it, and the IP address of the remote peer is not a configured one, the local system creates a temporary peer entry, completes initialization, sends an OPEN message to its peer, sets its Hold Timer to a large value, and changes its state to OpenSent.

If the local system does not allow BGP connections with unconfigured peers, then the local system rejects connections from IP addresses that are not configured peers, and remains in the Active state.

Start event is ignored in the Active state.

In response to any other event (initiated by either system or operator), the local system releases all BGP resources associated with this connection and changes its state to Idle.

OpenSent state:

In this state BGP waits for an OPEN message from its peer.

When an OPEN message is received, all fields are checked for correctness. If the BGP message header checking or OPEN message checking detects an error (see Section 6.2), or a connection collision (see Section 6.8) the local system sends a NOTIFICATION message and changes its state to Idle.

If there are no errors in the OPEN message, BGP sends a KEEPALIVE message and sets a KeepAlive timer. The Hold Timer, which was originally set to a large value (see above), is replaced with the negotiated Hold Time value (see section 4.2). If the negotiated Hold Time value is zero, then the Hold Time timer and KeepAlive timers are not started. If the value of the Autonomous System field is the same as the local Autonomous System number, then the connection is an "internal" connection; otherwise, it is "external". (This will effect UPDATE processing as described below.) Finally, the state is changed to OpenConfirm.

If a disconnect notification is received from the underlying transport protocol, the local system closes the BGP connection, restarts the ConnectRetry timer, while continue listening for connection that may be initiated by the remote BGP peer, and goes into the Active state.

If the Hold Timer expires, the local system sends NOTIFICATION message with error code Hold Timer Expired and changes its state to Idle.

In response to the Stop event (initiated by either system or

operator) the local system sends NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the OpenSent state.

In response to any other event the local system sends NOTIFICATION message with Error Code Finite State Machine Error and changes its state to Idle.

Whenever BGP changes its state from OpenSent to Idle, it closes the BGP (and transport-level) connection and releases all resources associated with that connection.

OpenConfirm state:

In this state BGP waits for a KEEPALIVE or NOTIFICATION message.

If the local system receives a KEEPALIVE message, it changes its state to Established.

If the Hold Timer expires before a KEEPALIVE message is received, the local system sends NOTIFICATION message with error code Hold Timer Expired and changes its state to Idle.

If the local system receives a NOTIFICATION message, it changes its state to Idle.

If the KeepAlive timer expires, the local system sends a KEEPALIVE message and restarts its KeepAlive timer.

If a disconnect notification is received from the underlying transport protocol, the local system changes its state to Idle.

In response to the Stop event (initiated by either system or operator) the local system sends NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the OpenConfirm state.

In response to any other event the local system sends NOTIFICATION message with Error Code Finite State Machine Error and changes its state to Idle.

Whenever BGP changes its state from OpenConfirm to Idle, it closes the BGP (and transport-level) connection and releases all resources associated with that connection.

Established state:

In the Established state BGP can exchange UPDATE, NOTIFICATION, and KEEPALIVE messages with its peer.

If the local system receives an UPDATE or KEEPALIVE message, it restarts its Hold Timer, if the negotiated Hold Time value is non-zero.

If the local system receives a NOTIFICATION message, it changes its state to Idle.

If the local system receives an UPDATE message and the UPDATE message error handling procedure (see Section 6.3) detects an error, the local system sends a NOTIFICATION message and changes its state to Idle.

If a disconnect notification is received from the underlying transport protocol, the local system changes its state to Idle.

If the Hold Timer expires, the local system sends a NOTIFICATION message with Error Code Hold Timer Expired and changes its state to Idle.

If the KeepAlive timer expires, the local system sends a KEEPALIVE message and restarts its KeepAlive timer.

Each time the local system sends a KEEPALIVE or UPDATE message, it restarts its KeepAlive timer, unless the negotiated Hold

Time value is zero.

In response to the Stop event (initiated by either system or operator), the local system sends a NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the Established state.

In response to any other event, the local system sends NOTIFICATION message with Error Code Finite State Machine Error
and changes its state to Idle.

Whenever BGP changes its state from Established to Idle, it closes the BGP (and transport-level) connection, releases all resources associated with that connection, and deletes all

routes derived from that connection.

9. UPDATE Message Handling

An UPDATE message may be received only in the Established state. When an UPDATE message is received, each field is checked for validity as specified in Section 6.3.

If an optional non-transitive attribute is unrecognized, it is quietly ignored. If an optional transitive attribute is unrecognized, the Partial bit (the third high-order bit) in the attribute flags octet is set to 1, and the attribute is retained for
propagation to other BGP speakers.

If an optional attribute is recognized, and has a valid value, then,
depending on the type of the optional attribute, it is processed

locally, retained, and updated, if necessary, for possible propagation to other BGP speakers.

If the UPDATE message contains a non-empty WITHDRAWN ROUTES field, the previously advertised routes whose destinations (expressed as IP prefixes) contained in this field shall be removed from the Adj-RIB-In. This BGP speaker shall run its Decision Process since the previously advertised route is not longer available for use.

If the UPDATE message contains a feasible route, it shall be placed in the appropriate Adj-RIB-In, and the following additional actions shall be taken:

i) If its Network Layer Reachability Information (NLRI) is identical to the one of a route currently stored in the Adj-RIB-In, then the new route shall replace the older route in the Adj-RIB-In, thus implicitly withdrawing the older route from service. The BGP speaker shall run its Decision Process since the older route is no longer available for use.

ii) If the new route is an overlapping route that is included (see 9.1.4) in an earlier route contained in the Adj-RIB-In, the BGP speaker shall run its Decision Process since the more specific route has implicitly made a portion of the less specific route unavailable for use.

iii) If the new route has identical path attributes to an earlier route contained in the Adj-RIB-In, and is more specific (see 9.1.4) than the earlier route, no further actions are necessary.

iv) If the new route has NLRI that is not present in any of the routes currently stored in the Adj-RIB-In, then the new route shall be placed in the Adj-RIB-In. The BGP speaker shall run its Decision Process.

v) If the new route is an overlapping route that is less specific (see 9.1.4) than an earlier route contained in the Adj-RIB-In, the BGP speaker shall run its Decision Process on the set of destinations described only by the less specific route.

9.1 Decision Process

The Decision Process selects routes for subsequent advertisement by applying the policies in the local Policy Information Base (PIB) to the routes stored in its Adj-RIB-In. The output of the Decision Process is the set of routes that will be advertised to all peers; the selected routes will be stored in the local speaker's Adj-RIB-Out.

The selection process is formalized by defining a function that takes the attribute of a given route as an argument and returns a non-negative integer denoting the degree of preference for the route. The function that calculates the degree of preference for a given route shall not use as its inputs any of the following: the existence of other routes, the non-existence of other routes, or the path attributes of other routes. Route selection then consists of individual application of the degree of preference function to each feasible route, followed by the choice of the one with the highest degree of preference.

The Decision Process operates on routes contained in each Adj-RIB-In, and is responsible for:

- selection of routes to be advertised to internal peers

- selection of routes to be advertised to external peers

- route aggregation and route information reduction

The Decision Process takes place in three distinct phases, each triggered by a different event:

a) Phase 1 is responsible for calculating the degree of preference

for each route received from an external peer, and MAY also advertise to all the internal peers the routes from external peers that have the highest degree of preference for each distinct

destination.

b) Phase 2 is invoked on completion of phase 1. It is responsible for choosing the best route out of all those available for each distinct destination, and for installing each chosen route into the appropriate Loc-RIB.

c) Phase 3 is invoked after the Loc-RIB has been modified. It is responsible for disseminating routes in the Loc-RIB to each external peer, according to the policies contained in the PIB. Route aggregation and information reduction can optionally be performed within this phase.

9.1.1 Phase 1: Calculation of Degree of Preference

The Phase 1 decision function shall be invoked whenever the local BGP speaker receives from a peer an UPDATE message that advertises a new route, a replacement route, or withdrawn routes.

The Phase 1 decision function is a separate process which completes when it has no further work to do.

The Phase 1 decision function shall lock an Adj-RIB-In prior to operating on any route contained within it, and shall unlock it after operating on all new or unfeasible routes contained within it.

For the newly received or replacement feasible route, the local BGP speaker shall determine a degree of preference. If the route is learned from an internal peer, the value of the LOCAL_PREF attribute shall be taken as the degree of preference. If the route is learned from an external peer, then the degree of preference shall be computed based on preconfigured policy information and used as the LOCAL_PREF value in any IBGP readvertisement. The exact nature of this policy information and the computation involved is a local matter. For a route learned from an external peer, the local speaker shall then run the internal update process of 9.2.1 to select and advertise the most preferable route.

9.1.2 Phase 2: Route Selection

The Phase 2 decision function shall be invoked on completion of Phase 1. The Phase 2 function is a separate process which completes when it has no further work to do. The Phase 2 process shall consider all routes that are present in the Adj-RIBs-In, including those received

from both internal and external peers.

The Phase 2 decision function shall be blocked from running while the Phase 3 decision function is in process. The Phase 2 function shall lock all Adj-RIBs-In prior to commencing its function, and shall unlock them on completion.

If the NEXT_HOP attribute of a BGP route depicts an address to which

the local BGP speaker doesn't have a route in its Loc-RIB, the BGP route should be excluded from the Phase 2 decision function.

It is critical that routers within an AS do not make conflicting decisions regarding route selection that would cause forwarding loops to occur.

For each set of destinations for which a feasible route exists in the Adj-RIBs-In, the local BGP speaker shall identify the route that has:

a) the highest degree of preference of any route to the same set of destinations, or

b) is the only route to that destination, or

c) is selected as a result of the Phase 2 tie breaking rules specified in 9.1.2.1.

The local speaker SHALL then install that route in the Loc-RIB, replacing any route to the same destination that is currently being held in the Loc-RIB. The local speaker MUST determine the immediate next hop to the address depicted by the NEXT_HOP attribute of the selected route by performing a lookup in the IGP and selecting one of the possible paths in the IGP. This immediate next hop MUST be used when installing the selected route in the Loc-RIB. If the route to the address depicted by the NEXT_HOP attribute changes such that the immediate next hop changes, route selection should be recalculated as specified above.

Unfeasible routes shall be removed from the Loc-RIB, and corresponding unfeasible routes shall then be removed from the Adj-RIBs-In.

9.1.2.1 Breaking Ties (Phase 2)

In its Adj-RIBs-In a BGP speaker may have several routes to the same destination that have the same degree of preference. The local

speaker can select only one of these routes for inclusion in the associated Loc-RIB. The local speaker considers all routes with the same degrees of preference, both those received from internal peers, and those received from external peers.

The following tie-breaking procedure assumes that for each candidate route all the BGP speakers within an autonomous system can ascertain the cost of a path (interior distance) to the address depicted by the NEXT_HOP attribute of the route.

The tie-breaking algorithm begins by considering all equally preferable routes and then selects routes to be removed from consideration. The algorithm terminates as soon as only one route remains in consideration. The criteria must be applied in the order specified.

Several of the criteria are described using pseudo-code. Note that the pseudo-code shown was chosen for clarity, not efficiency. It is not intended to specify any particular implementation. BGP implementations MAY use any algorithm which produces the same results as those described here.

a) Remove from consideration routes with less-preferred MULTI_EXIT_DISC attributes. MULTI_EXIT_DISC is only comparable between routes learned from the same neighboring AS. Routes which do not have the MULTI_EXIT_DISC attribute are considered to have the highest possible MULTI_EXIT_DISC value.

This is also described in the following procedure:

```
            for m = all routes still under consideration
                for n = all routes still under consideration
                    if (neighborAS(m) == neighborAS(n)) and (MED(n) <
MED(m))
                        remove route m from consideration
```

In the pseudo-code above, MED(n) is a function which returns the value of route n's MULTI_EXIT_DISC attribute. If route n has no MULTI_EXIT_DISC attribute, the function returns the highest possible MULTI_EXIT_DISC value, i.e. 2^32-1.

Similarly, neighborAS(n) is a function which returns the neighbor AS from which the route was received.

b) Remove from consideration any routes with less-preferred interior cost. The interior cost of a route is determined by calculating the metric to the next hop for the route using the interior routing protocol(s). If the next hop for a route is reachable, but no cost can be determined, then this step should be

should be skipped (equivalently, consider all routes to have equal costs).

This is also described in the following procedure.

```
            for m = all routes still under consideration
                for n = all routes in still under consideration
                    if (cost(n) is better than cost(m))
                        remove m from consideration
```

In the pseudo-code above, cost(n) is a function which returns the cost of the path (interior distance) to the address given in the NEXT_HOP attribute of the route.

c) If at least one of the candidate routes was received from an external peer in a neighboring autonomous system, remove from consideration all routes which were received from internal peers.

d) Remove from consideration all routes other than the route that was advertised by the BGP speaker whose BGP Identifier has the lowest value.

9.1.3 Phase 3: Route Dissemination

The Phase 3 decision function shall be invoked on completion of Phase 2, or when any of the following events occur:

a) when routes in a Loc-RIB to local destinations have changed

b) when locally generated routes learned by means outside of BGP have changed

c) when a new BGP speaker – BGP speaker connection has been established

The Phase 3 function is a separate process which completes when it has no further work to do. The Phase 3 Routing Decision function shall be blocked from running while the Phase 2 decision function is in process.

All routes in the Loc-RIB shall be processed into a corresponding entry in the associated Adj-RIBs-Out. Route aggregation and information reduction techniques (see 9.2.4.1) may optionally be applied.

For the benefit of future support of inter-AS multicast capabilities,

a BGP speaker that participates in inter-AS multicast routing shall advertise a route it receives from one of its external peers and if it installs it in its Loc-RIB, it shall advertise it back to the peer from which the route was received. For a BGP speaker that does not participate in inter-AS multicast routing such an advertisement is optional. When doing such an advertisement, the NEXT_HOP attribute should be set to the address of the peer. An implementation may also optimize such an advertisement by truncating information in the AS_PATH attribute to include only its own AS number and that of the peer that advertised the route (such truncation requires the ORIGIN attribute to be set to INCOMPLETE). In addition an implementation is not required to pass optional or discretionary path attributes with such an advertisement.

When the updating of the Adj-RIBs-Out and the Forwarding Information Base (FIB) is complete, the local BGP speaker shall run the external update process of 9.2.2.

9.1.4 Overlapping Routes

A BGP speaker may transmit routes with overlapping Network Layer Reachability Information (NLRI) to another BGP speaker. NLRI overlap occurs when a set of destinations are identified in non-matching multiple routes. Since BGP encodes NLRI using IP prefixes, overlap will always exhibit subset relationships. A route describing a smaller set of destinations (a longer prefix) is said to be more specific than a route describing a larger set of destinations (a shorted prefix); similarly, a route describing a larger set of destinations (a shorter prefix) is said to be less specific than a route describing a smaller set of destinations (a longer prefix).

The precedence relationship effectively decomposes less specific routes into two parts:

- a set of destinations described only by the less specific route, and

- a set of destinations described by the overlap of the less

specific and the more specific routes

When overlapping routes are present in the same Adj-RIB-In, the more
specific route shall take precedence, in order from more specific to
least specific.

The set of destinations described by the overlap represents a portion

of the less specific route that is feasible, but is not currently in
use. If a more specific route is later withdrawn, the set of
destinations described by the overlap will still be reachable using
the less specific route.

If a BGP speaker receives overlapping routes, the Decision Process
MUST consider both routes based on the configured acceptance policy.
If both a less and a more specific route are accepted, then the
Decision Process MUST either install both the less and the more
specific routes or it MUST aggregate the two routes and install the
aggregated route.

If a BGP speaker chooses to aggregate, then it MUST add
ATOMIC_AGGREGATE attribute to the route. A route that carries
ATOMIC_AGGREGATE attribute can not be de-aggregated. That is, the
NLRI of this route can not be made more specific. Forwarding along
such a route does not guarantee that IP packets will actually
traverse only ASs listed in the AS_PATH attribute of the route.

9.2 Update-Send Process

The Update-Send process is responsible for advertising UPDATE messages to all peers. For example, it distributes the routes chosen by the Decision Process to other BGP speakers which may be located in either the same autonomous system or a neighboring autonomous system.

Rules for information exchange between BGP speakers located in different autonomous systems are given in 9.2.2; rules for information exchange between BGP speakers located in the same autonomous system are given in 9.2.1.

Distribution of routing information between a set of BGP speakers, all of which are located in the same autonomous system, is referred to as internal distribution.

9.2.1 Internal Updates

The Internal update process is concerned with the distribution of routing information to internal peers.

When a BGP speaker receives an UPDATE message from an internal peer, the receiving BGP speaker shall not re-distribute the routing information contained in that UPDATE message to other internal peers.

When a BGP speaker receives a new route from an external peer, it

MUST advertise that route to all other internal peers by means of an UPDATE message if this route will be installed in its Loc-RIB according to the route selection rules in 9.1.2.

When a BGP speaker receives an UPDATE message with a non-empty

WITHDRAWN ROUTES field, it shall remove from its Adj-RIB-In all routes whose destinations was carried in this field (as IP prefixes).

The speaker shall take the following additional steps:

1) if the corresponding feasible route had not been previously advertised, then no further action is necessary

2) if the corresponding feasible route had been previously advertised, then:

i) if a new route is selected for advertisement that has the same Network Layer Reachability Information as the unfeasible routes, then the local BGP speaker shall advertise the replacement route

ii) if a replacement route is not available for advertisement, then the BGP speaker shall include the destinations of the unfeasible route (in form of IP prefixes) in the WITHDRAWN ROUTES field of an UPDATE message, and shall send this message to each peer to whom it had previously advertised the corresponding feasible route.

All feasible routes which are advertised shall be placed in the appropriate Adj-RIBs-Out, and all unfeasible routes which are advertised shall be removed from the Adj-RIBs-Out.

9.2.1.1 Breaking Ties (Internal Updates)

If a local BGP speaker has connections to several external peers, there will be multiple Adj-RIBs-In associated with these peers. These Adj-RIBs-In might contain several equally preferable routes to the same destination, all of which were advertised by external peers. The local BGP speaker shall select one of these routes according to the following rules:

a) If the candidate routes differ only in their NEXT_HOP and MULTI_EXIT_DISC attributes, and the local system is configured to take into account the MULTI_EXIT_DISC attribute, select the route that has the lowest value of the MULTI_EXIT_DISC attribute. A route with the MULTI_EXIT_DISC attribute shall be preferred to a

route without the MULTI_EXIT_DISC attribute.

b) If the local system can ascertain the cost of a path to the entity depicted by the NEXT_HOP attribute of the candidate route, select the route with the lowest cost.

c) In all other cases, select the route that was advertised by the BGP speaker whose BGP Identifier has the lowest value.

9.2.2 External Updates

The external update process is concerned with the distribution of routing information to external peers. As part of Phase 3 route selection process, the BGP speaker has updated its Adj-RIBs-Out and its Forwarding Table. All newly installed routes and all newly unfeasible routes for which there is no replacement route shall be advertised to external peers by means of UPDATE message.

Any routes in the Loc-RIB marked as unfeasible shall be removed. Changes to the reachable destinations within its own autonomous system shall also be advertised in an UPDATE message.

9.2.3 Controlling Routing Traffic Overhead

The BGP protocol constrains the amount of routing traffic (that is, UPDATE messages) in order to limit both the link bandwidth needed to advertise UPDATE messages and the processing power needed by the Decision Process to digest the information contained in the UPDATE messages.

9.2.3.1 Frequency of Route Advertisement

The parameter MinRouteAdvertisementInterval determines the minimum amount of time that must elapse between advertisement of routes to a particular destination from a single BGP speaker. This rate limiting procedure applies on a per-destination basis, although the value of MinRouteAdvertisementInterval is set on a per BGP peer basis.

Two UPDATE messages sent from a single BGP speaker that advertise feasible routes to some common set of destinations received from external peers must be separated by at least

MinRouteAdvertisementInterval. Clearly, this can only be achieved precisely by keeping a separate timer for each common set of destinations. This would be unwarranted overhead. Any technique which ensures that the interval between two UPDATE messages sent from a single BGP speaker that advertise feasible routes to some common set of destinations received from external peers will be at least MinRouteAdvertisementInterval, and will also ensure a constant upper bound on the interval is acceptable.

Since fast convergence is needed within an autonomous system, this procedure does not apply for routes received from other internal peers. To avoid long-lived black holes, the procedure does not apply to the explicit withdrawal of unfeasible routes (that is, routes whose destinations (expressed as IP prefixes) are listed in the WITHDRAWN ROUTES field of an UPDATE message).

This procedure does not limit the rate of route selection, but only the rate of route advertisement. If new routes are selected multiple times while awaiting the expiration of MinRouteAdvertisementInterval, the last route selected shall be advertised at the end of MinRouteAdvertisementInterval.

9.2.3.2 Frequency of Route Origination

The parameter MinASOriginationInterval determines the minimum amount of time that must elapse between successive advertisements of UPDATE messages that report changes within the advertising BGP speaker's own autonomous systems.

9.2.3.3 Jitter

To minimize the likelihood that the distribution of BGP messages by a given BGP speaker will contain peaks, jitter should be applied to the timers associated with MinASOriginationInterval, Keepalive, and MinRouteAdvertisementInterval. A given BGP speaker shall apply the same jitter to each of these quantities regardless of the destinations to which the updates are being sent; that is, jitter will not be applied on a "per peer" basis.

The amount of jitter to be introduced shall be determined by multiplying the base value of the appropriate timer by a random factor which is uniformly distributed in the range from 0.75 to 1.0.

9.2.4 Efficient Organization of Routing Information

Having selected the routing information which it will advertise, a BGP speaker may avail itself of several methods to organize this information in an efficient manner.

9.2.4.1 Information Reduction

Information reduction may imply a reduction in granularity of policy control - after information is collapsed, the same policies will apply to all destinations and paths in the equivalence class.

The Decision Process may optionally reduce the amount of information that it will place in the Adj-RIBs-Out by any of the following methods:

a) Network Layer Reachability Information (NLRI):

Destination IP addresses can be represented as IP address prefixes. In cases where there is a correspondence between the address structure and the systems under control of an autonomous system administrator, it will be possible to reduce the size of the NLRI carried in the UPDATE messages.

b) AS_PATHs:

AS path information can be represented as ordered AS_SEQUENCEs or unordered AS_SETs. AS_SETs are used in the route aggregation algorithm described in 9.2.4.2. They reduce the size of the AS_PATH information by listing each AS number only once, regardless of how many times it may have appeared in multiple AS_PATHs that were aggregated.

An AS_SET implies that the destinations listed in the NLRI can be reached through paths that traverse at least some of the constituent autonomous systems. AS_SETs provide sufficient information to avoid routing information looping; however their use may prune potentially feasible paths, since such paths are no

longer listed individually as in the form of AS_SEQUENCEs. In practice this is not likely to be a problem, since once an IP packet arrives at the edge of a group of autonomous systems, the BGP speaker at that point is likely to have more detailed path information and can distinguish individual paths to destinations.

9.2.4.2 Aggregating Routing Information

Aggregation is the process of combining the characteristics of several different routes in such a way that a single route can be advertised. Aggregation can occur as part of the decision process to reduce the amount of routing information that will be placed in the Adj-RIBs-Out.

Aggregation reduces the amount of information that a BGP speaker must store and exchange with other BGP speakers. Routes can be aggregated by applying the following procedure separately to path attributes of like type and to the Network Layer Reachability Information.

Routes that have the following attributes shall not be aggregated unless the corresponding attributes of each route are identical: MULTI_EXIT_DISC, NEXT_HOP.

Path attributes that have different type codes can not be aggregated together. Path of the same type code may be aggregated, according to the following rules:

ORIGIN attribute: If at least one route among routes that are

aggregated has ORIGIN with the value INCOMPLETE, then the aggregated route must have the ORIGIN attribute with the value INCOMPLETE. Otherwise, if at least one route among routes that are aggregated has ORIGIN with the value EGP, then the aggregated route must have the origin attribute with the value EGP. In all other case the value of the ORIGIN attribute of the aggregated route is INTERNAL.

AS_PATH attribute: If routes to be aggregated have identical AS_PATH attributes, then the aggregated route has the same AS_PATH attribute as each individual route.

For the purpose of aggregating AS_PATH attributes we model each AS within the AS_PATH attribute as a tuple <type, value>, where "type" identifies a type of the path segment the AS belongs to (e.g. AS_SEQUENCE, AS_SET), and "value" is the AS number. If the routes to be aggregated have different AS_PATH attributes, then the aggregated AS_PATH attribute shall satisfy all of the following conditions:

- all tuples of the type AS_SEQUENCE in the aggregated AS_PATH shall appear in all of the AS_PATH in the initial set of routes to be aggregated.

- all tuples of the type AS_SET in the aggregated AS_PATH shall

appear in at least one of the AS_PATH in the initial set (they may appear as either AS_SET or AS_SEQUENCE types).

- for any tuple X of the type AS_SEQUENCE in the aggregated AS_PATH which precedes tuple Y in the aggregated AS_PATH, X precedes Y in each AS_PATH in the initial set which contains Y, regardless of the type of Y.

- No tuple with the same value shall appear more than once in the aggregated AS_PATH, regardless of the tuple's type.

An implementation may choose any algorithm which conforms to these rules. At a minimum a conformant implementation shall be able to perform the following algorithm that meets all of the above conditions:

- determine the longest leading sequence of tuples (as defined above) common to all the AS_PATH attributes of the routes to be aggregated. Make this sequence the leading sequence of the aggregated AS_PATH attribute.

- set the type of the rest of the tuples from the AS_PATH attributes of the routes to be aggregated to AS_SET, and append them to the aggregated AS_PATH attribute.

- if the aggregated AS_PATH has more than one tuple with the same value (regardless of tuple's type), eliminate all, but one such tuple by deleting tuples of the type AS_SET from the aggregated AS_PATH attribute.

Appendix 6, section 6.8 presents another algorithm that satisfies the conditions and allows for more complex policy configurations.

ATOMIC_AGGREGATE: If at least one of the routes to be aggregated has ATOMIC_AGGREGATE path attribute, then the aggregated route shall have this attribute as well.

AGGREGATOR: All AGGREGATOR attributes of all routes to be aggregated should be ignored.

9.3 Route Selection Criteria

Generally speaking, additional rules for comparing routes among several alternatives are outside the scope of this document. There are two exceptions:

- If the local AS appears in the AS path of the new route being considered, then that new route cannot be viewed as better than any other route. If such a route were ever used, a routing loop could result (see Section 6.3).

- In order to achieve successful distributed operation, only routes with a likelihood of stability can be chosen. Thus, an AS must avoid using unstable routes, and it must not make rapid spontaneous changes to its choice of route. Quantifying the terms "unstable" and "rapid" in the previous sentence will require experience, but the principle is clear.

9.4 Originating BGP routes

A BGP speaker may originate BGP routes by injecting routing information acquired by some other means (e.g. via an IGP) into BGP.
A BGP speaker that originates BGP routes shall assign the degree of preference to these routes by passing them through the Decision Process (see Section 9.1). These routes may also be distributed to other BGP speakers within the local AS as part of the Internal update process (see Section 9.2.1). The decision whether to distribute non-BGP acquired routes within an AS via BGP or not depends on the environment within the AS (e.g. type of IGP) and should be controlled via configuration.

Appendix 1. BGP FSM State Transitions and Actions.

This Appendix discusses the transitions between states in the BGP FSM in response to BGP events. The following is the list of these states and events when the negotiated Hold Time value is non-zero.

BGP States:

1 - Idle
2 - Connect
3 - Active
4 - OpenSent
5 - OpenConfirm
6 - Established

BGP Events:

1 - BGP Start
2 - BGP Stop
3 - BGP Transport connection open
4 - BGP Transport connection closed
5 - BGP Transport connection open failed
6 - BGP Transport fatal error
7 - ConnectRetry timer expired
8 - Hold Timer expired
9 - KeepAlive timer expired
10 - Receive OPEN message
11 - Receive KEEPALIVE message
12 - Receive UPDATE messages

13 – Receive NOTIFICATION message

The following table describes the state transitions of the BGP FSM and the actions triggered by these transitions.

Event	Actions	Message Sent	Next State
Idle (1)			
1	Initialize resources	none	2
	Start ConnectRetry timer Initiate a transport connection		
others	none	none	1
Connect(2)			
1	none	none	2
3	Complete initialization	OPEN	4
	Clear ConnectRetry timer		
5	Restart ConnectRetry timer	none	3
7	Restart ConnectRetry timer	none	2
	Initiate a transport connection		
others	Release resources	none	1
Active (3)			
1	none	none	3
3	Complete initialization	OPEN	4
	Clear ConnectRetry timer		
5	Close connection		3
	Restart ConnectRetry timer		
7	Restart ConnectRetry timer	none	2
	Initiate a transport connection		
others	Release resources	none	1

OpenSent(4)			
1	none	none	4
4	Close transport connection	none	3
	Restart ConnectRetry timer		
6	Release resources	none	1
10	Process OPEN is OK	KEEPALIVE	5
	Process OPEN failed	NOTIFICATION	1
others	Close transport connection	NOTIFICATION	1
	Release resources		
OpenConfirm (5)			
1	none	none	5
4	Release resources	none	1
6	Release resources	none	1
9	Restart KeepAlive timer	KEEPALIVE	5
11	Complete initialization	none	6
	Restart Hold Timer		
13	Close transport connection		1
	Release resources		
others	Close transport connection	NOTIFICATION	1
	Release resources		

Established (6)			
1	none	none	6
4	Release resources	none	1
6	Release resources	none	1
9	Restart KeepAlive timer	KEEPALIVE	6
11	Restart Hold Timer	KEEPALIVE	6
12	Process UPDATE is OK	UPDATE	6
	Process UPDATE failed	NOTIFICATION	1
13	Close transport connection		1
	Release resources		
others	Close transport connection	NOTIFICATION	1
	Release resources		

The following is a condensed version of the above state transition table.

Events	Idle	Connect	Active	OpenSent	OpenConfirm	Estab

	(1)	(2)	(3)	(4)	(5)	(6)
1	2	2	3	4	5	6
2	1	1	1	1	1	1
3	1	4	4	1	1	1
4	1	1	1	3	1	1
5	1	3	3	1	1	1
6	1	1	1	1	1	1
7	1	2	2	1	1	1
8	1	1	1	1	1	1
9	1	1	1	1	5	6
10	1	1	1	1 or 5	1	1
11	1	1	1	1	6	6
12	1	1	1	1	1	1 or 6
13	1	1	1	1	1	1

Appendix 2. Comparison with RFC1267

BGP-4 is capable of operating in an environment where a set of reachable destinations may be expressed via a single IP prefix. The concept of network classes, or subnetting is foreign to BGP-4. To accommodate these capabilities BGP-4 changes semantics and encoding associated with the AS_PATH attribute. New text has been added to define semantics associated with IP prefixes. These abilities allow BGP-4 to support the proposed supernetting scheme [9].

To simplify configuration this version introduces a new attribute, LOCAL_PREF, that facilitates route selection procedures.

The INTER_AS_METRIC attribute has been renamed to be MULTI_EXIT_DISC.

A new attribute, ATOMIC_AGGREGATE, has been introduced to insure that certain aggregates are not de-aggregated. Another new attribute, AGGREGATOR, can be added to aggregate routes in order to advertise which AS and which BGP speaker within that AS caused the aggregation.

To insure that Hold Timers are symmetric, the Hold Time is now negotiated on a per-connection basis. Hold Times of zero are now supported.

Appendix 3. Comparison with RFC 1163

All of the changes listed in Appendix 2, plus the following.

To detect and recover from BGP connection collision, a new field (BGP Identifier) has been added to the OPEN message. New text (Section 6.8) has been added to specify the procedure for detecting and recovering from collision.

The new document no longer restricts the border router that is passed in the NEXT_HOP path attribute to be part of the same Autonomous System as the BGP Speaker.

New document optimizes and simplifies the exchange of the information about previously reachable routes.

Appendix 4. Comparison with RFC 1105

All of the changes listed in Appendices 2 and 3, plus the following.

Minor changes to the RFC1105 Finite State Machine were necessary to accommodate the TCP user interface provided by 4.3 BSD.

The notion of Up/Down/Horizontal relations present in RFC1105 has been removed from the protocol.

The changes in the message format from RFC1105 are as follows:

1. The Hold Time field has been removed from the BGP header and added to the OPEN message.

2. The version field has been removed from the BGP header and added to the OPEN message.

3. The Link Type field has been removed from the OPEN message.

4. The OPEN CONFIRM message has been eliminated and replaced with implicit confirmation provided by the KEEPALIVE message.

5. The format of the UPDATE message has been changed significantly. New fields were added to the UPDATE message to support multiple path attributes.

6. The Marker field has been expanded and its role broadened to support authentication.

Note that quite often BGP, as specified in RFC 1105, is referred to as BGP-1, BGP, as specified in RFC 1163, is referred to as BGP-2, BGP, as specified in RFC1267 is referred to as BGP-3, and BGP, as specified in this document is referred to as BGP-4.

Appendix 5. TCP options that may be used with BGP

If a local system TCP user interface supports TCP PUSH function, then each BGP message should be transmitted with PUSH flag set. Setting PUSH flag forces BGP messages to be transmitted promptly to the receiver.

If a local system TCP user interface supports setting precedence for TCP connection, then the BGP transport connection should be opened with precedence set to Internetwork Control (110) value (see also [6]).

Appendix 6. Implementation Recommendations

This section presents some implementation recommendations.

6.1 Multiple Networks Per Message

The BGP protocol allows for multiple address prefixes with the same AS path and next-hop gateway to be specified in one message. Making use of this capability is highly recommended. With one address prefix per message there is a substantial increase in overhead in the receiver. Not only does the system overhead increase due to the reception of multiple messages, but the overhead of scanning the routing table for updates to BGP peers and other routing protocols (and sending the associated messages) is incurred multiple times as

well. One method of building messages containing many address prefixes per AS path and gateway from a routing table that is not organized per AS path is to build many messages as the routing table is scanned. As each address prefix is processed, a message for the associated AS path and gateway is allocated, if it does not exist, and the new address prefix is added to it. If such a message exists, the new address prefix is just appended to it. If the message lacks the space to hold the new address prefix, it is transmitted, a new message is allocated, and the new address prefix is inserted into the new message. When the entire routing table has been scanned, all allocated messages are sent and their resources released. Maximum compression is achieved when all the destinations covered by the address prefixes share a gateway and common path attributes, making it possible to send many address prefixes in one 4096-byte message.

When peering with a BGP implementation that does not compress multiple address prefixes into one message, it may be necessary to take steps to reduce the overhead from the flood of data received when a peer is acquired or a significant network topology change occurs. One method of doing this is to limit the rate of updates. This will eliminate the redundant scanning of the routing table to provide flash updates for BGP peers and other routing protocols. A disadvantage of this approach is that it increases the propagation latency of routing information. By choosing a minimum flash update interval that is not much greater than the time it takes to process the multiple messages this latency should be minimized. A better method would be to read all received messages before sending updates.

6.2 Processing Messages on a Stream Protocol

BGP uses TCP as a transport mechanism. Due to the stream nature of TCP, all the data for received messages does not necessarily arrive at the same time. This can make it difficult to process the data as messages, especially on systems such as BSD Unix where it is not possible to determine how much data has been received but not yet processed.

One method that can be used in this situation is to first try to read just the message header. For the KEEPALIVE message type, this is a complete message; for other message types, the header should first be verified, in particular the total length. If all checks are

successful, the specified length, minus the size of the message header is the amount of data left to read. An implementation that would "hang" the routing information process while trying to read from a peer could set up a message buffer (4096 bytes) per peer and fill it with data as available until a complete message has been

received.

6.3 Reducing route flapping

To avoid excessive route flapping a BGP speaker which needs to withdraw a destination and send an update about a more specific or less specific route SHOULD combine them into the same UPDATE message.

6.4 BGP Timers

BGP employs five timers: ConnectRetry, Hold Time, KeepAlive, MinASOriginationInterval, and MinRouteAdvertisementInterval The suggested value for the ConnectRetry timer is 120 seconds. The suggested value for the Hold Time is 90 seconds. The suggested value for the KeepAlive timer is 30 seconds. The suggested value for the MinASOriginationInterval is 15 seconds. The suggested value for the MinRouteAdvertisementInterval is 30 seconds.

An implementation of BGP MUST allow these timers to be configurable.

6.5 Path attribute ordering

Implementations which combine update messages as described above in 6.1 may prefer to see all path attributes presented in a known order.
This permits them to quickly identify sets of attributes from different update messages which are semantically identical. To facilitate this, it is a useful optimization to order the path attributes according to type code. This optimization is entirely optional.

6.6 AS_SET sorting

Another useful optimization that can be done to simplify this situation is to sort the AS numbers found in an AS_SET. This optimization is entirely optional.

6.7 Control over version negotiation

Since BGP-4 is capable of carrying aggregated routes which cannot be properly represented in BGP-3, an implementation which supports BGP-4 and another BGP version should provide the capability to only speak BGP-4 on a per-peer basis.

6.8 Complex AS_PATH aggregation

An implementation which chooses to provide a path aggregation algorithm which retains significant amounts of path information may wish to use the following procedure:

For the purpose of aggregating AS_PATH attributes of two routes, we model each AS as a tuple <type, value>, where "type" identifies a type of the path segment the AS belongs to (e.g. AS_SEQUENCE, AS_SET), and "value" is the AS number. Two ASs are said to be the same if their corresponding <type, value> tuples are the same.

The algorithm to aggregate two AS_PATH attributes works as follows:

a) Identify the same ASs (as defined above) within each AS_PATH attribute that are in the same relative order within both AS_PATH attributes. Two ASs, X and Y, are said to be in the same order if either:
- X precedes Y in both AS_PATH attributes, or - Y precedes X in both AS_PATH attributes.

b) The aggregated AS_PATH attribute consists of ASs identified in (a) in exactly the same order as they appear in the AS_PATH attributes to be aggregated. If two consecutive ASs identified in (a) do not immediately follow each other in both of the AS_PATH attributes to be aggregated, then the intervening ASs (ASs that are between the two consecutive ASs that are the same) in both attributes are combined into an AS_SET path segment that consists of the intervening ASs from both AS_PATH attributes; this segment is then placed in between the two consecutive ASs identified in (a) of the aggregated attribute. If two consecutive ASs identified in (a) immediately follow each other in one attribute, but do not follow in another, then the intervening ASs of the latter are combined into an AS_SET path segment; this segment is then placed in between the two consecutive ASs identified in (a) of the aggregated attribute.

If as a result of the above procedure a given AS number appears more than once within the aggregated AS_PATH attribute, all, but the last instance (rightmost occurrence) of that AS number should be removed from the aggregated AS_PATH attribute.

References

[1] Mills, D., "Exterior Gateway Protocol Formal Specification", RFC904, April 1984.

[2] Rekhter, Y., "EGP and Policy Based Routing in the New NSFNET Backbone", RFC1092, February 1989.

[3] Braun, H-W., "The NSFNET Routing Architecture", RFC1093, February 1989.

[4] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC793, September 1981.

[5] Rekhter, Y., and P. Gross, "Application of the Border Gateway Protocol in the Internet", RFC1772, March 1995.

[6] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", RFC791, September 1981.

[7] "Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol for Exchange of Inter-domain Routeing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs", ISO/IEC IS10747, 1993

[8] Fuller, V., Li, T., Yu, J., and Varadhan, K., ""Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC1519, September 1993.

[9] Rekhter, Y., Li, T., "An Architecture for IP Address Allocation with CIDR", RFC 1518, September 1993.

Security Considerations

Security issues are not discussed in this document.

Editors' Addresses

Yakov Rekhter
cisco Systems, Inc.

170 W. Tasman Dr.
San Jose, CA 95134
email: yakov@cisco.com

Tony Li
Procket Networks
3910 Freedom Circle, Ste. 102A
Santa Clara CA 95054
Email: tli@procket.com

Network Working Group
Henk Smit
INTERNET DRAFT Cisco Systems

Tony Li
Juniper Networks

May 1999

IS-IS extensions for Traffic Engineering

<draft-ietf-isis-traffic-01.txt>

Status

1.0 Abstract

This document describes extensions to the IS-IS protocol to support Traffic Engineering [1]. The IS-IS protocol is specified in [2], with extensions for supporting IPv4 specified in [3].

This document extends the IS-IS protocol by specifying new information that a Intermediate System (IS) [router] can place in Link State Protocol Data Units (LSPs). This information describes additional information about the state of the network that is useful

for traffic engineering computations.

2.0 Introduction

An IS-IS LSP is composed of a fixed header and a number of tuples, each consisting of a Type, a Length, and a Value. Such tuples are commonly known as TLVs, and are a good way of encoding information in a flexible and extensible format.

The changes in this document include the design of new TLVs to replace the existing IS Neighbor TLV, IP Reachability TLV and add additional information. Mechanisms and procedures to migrate to the new TLVs are not discussed in this document.

The primary goal of these extensions is to add more information about the characteristics of a particular link to an IS-IS's LSP. Secondary goals include increasing the dynamic range of the IS-IS metric and improving the encoding of IP prefixes. The router id is useful for traffic engineering purposes because it describes a single address that can always be used to reference a particular router.

This document is a publication of the IS-IS Working Group within the IETF, and is a contribution to ISO IEC JTC1/SC6, for eventual inclusion with ISO 10589.

3.0 The router ID TLV

The router ID TLV is TLV type 134.

The router ID TLV contains the 4-octet router ID of the router originating the LSP. This is useful in several regards:

For traffic engineering, it guarantees that we have a single stable address that can always be referenced in a path that will be reachable from multiple hops away, regardless of the state of the node's interfaces.

If OSPF is also active in the domain, traffic engineering can compute

the mapping between the OSPF and IS-IS topologies.

Implementations MUST NOT inject a /32 prefix for the router ID into their forwarding table, because this can lead to forwarding loops when interacting with systems that do not support this TLV.

4.0 The extended IP reachability TLV

The extended IP reachability TLV is TLV type 135.

The existing IP reachability TLV is a single TLV that carries IP prefixes in a format that is analogous to the IS neighbor TLV. It carries four metrics, of which only the default metric is commonly used. Of this, the default metric has a possible range of 0-63. This limitation is one of the restrictions that we would like to lift.

In addition, route redistribution (a.k.a. route leaking) is a key problem that is not addressed by the existing IP reachability TLV. This problem occurs when an IP prefix is injected into a level one area, redistributed into level 2, subsequently redistributed into a second level one area, and then redistributed from the second level one area back into level two. This problem occurs because the path that the information can take forms a loop. The likely result is a forwarding loop.

To address these issues, the proposed extended IP reachability TLV provides for a 32 bit metric and adds one bit to indicate that a prefix has been redistributed 'down' in the hierarchy.

The proposed extended IP reachability TLV contains a new data structure, consisting of:

- 4 bytes of metric information
- 1 byte of control information, consisting of
  - 1 bit of up/down information
  - 1 bit indicating the existence of sub-TLVs
  - 6 bits of prefix length
- 0-4 bytes of IPv4 prefix
- 0-250 optional octets of sub-TVLs, if present consisting of
  - 1 octet of length of sub-TLVs
  - 0-249 octets of sub-TLVs

This data structure can be replicated within the TLV, not to exceed the maximum length of the TLV.

The up/down bit shall be set to 0 when a prefix is first injected into IS-IS. If a prefix is redistributed from a higher level to a lower level (e.g. level two to level one), the bit shall be set to 1, to indicate that the prefix has travelled down the hierarchy. Prefixes that have the up/down bit set to 1 must not be redistributed. If a prefix is redistributed from an area to another area at the same level, then the up/down bit shall be set to 1.

These semantics apply even if IS-IS is extended in the future to have additional levels. By insuring that prefixes follow only the IS-IS hierarchy, we have insured that the information does not loop, thereby insuring that there are no persistent forwarding loops.

If there are no sub-TLVs associated with this IP prefix, the bit indicating the presence of sub-TVLs shall be set to 0. If this bit is set to 1, the first octet after the prefix will be interpreted as the length of sub-TLVs. Please note that while the encoding allows for 255 octets of sub-TLVs, the maximum value cannot fit in the overall extended IP reachability TLV. The practical maximum is 255 octets minus the 5-9 octets described above, or 250 octets. No sub-TLVs for the extended IP reachability TLV have been defined yet.

The 6 bits of prefix length can have the values 0-32 and indicate the number of significant bits in the prefix. The prefix is encoded in the minimal number of bytes for the given number of significant bits. This implies:

Significant bits	Bytes
0	0
1-8	1
9-16	2
17-24	3
25-32	4

The remaining bits of prefix are transmitted as zero and ignored upon

receipt.

If an IP prefix is advertised with a metric larger then MAX_PATH_METRIC (0xFE000000, see below), this IP prefix should not be considered during the normal SPF computation. This will allow advertisment of an IP prefix for other purposes than building the normal IP routing table.

5.0 The extended IS reachability TLV

The extended IS reachability TLV is TLV type 22.

The existing IS reachability TLV is a single TLV that contains information about a series of IS neighbors. For each neighbor, there is a structure that contains the default metric, the delay, the monetary cost, the reliability, and the 7-octet ID of the adjacent neighbor. Of this information, the default metric is commonly used.
The default metric is currently one octet, with one bit used to indicate that the metric is present and one bit used to indicate

whether the metric is internal or external. The remaining 6 bits are used to store the actual metric, resulting a possible metric range of 0-63. This limitation is one of the restrictions that we would like to lift.

The remaining three metrics (delay, monetary cost, and reliability) are not commonly implemented and reflect unused overhead in the TLV.
The neighbor is identified by its system Id (typically 6-octets), plus one octet to indicate the pseudonode number if the neighbor is on a LAN interface. Thus, the existing TLV consumes 11 octets per neighbor, with 4 octets for metric and 7 octets for neighbor identification. To indicate multiple adjacencies, this structure is repeated within the IS reachability TLV. Because the TLV is limited to 255 octets of content, a single TLV can describe up to 23 neighbors. The IS reachability TLV can be repeated within the LSP

fragments to describe further neighbors.

The proposed extended IS reachability TLV contains a new data structure, consisting of

- 7 octets of system Id and pseudonode number
- 3 octets of default metric
- 1 octet of length of sub-TLVs
- 0-244 octets of sub-TLVs

Thus, if no sub-TLVs are used, the new encoding requires 11 octets and can contain up to 23 neighbors. Please note that while the encoding allows for 255 octets of sub-TLVs, the maximum value cannot fit in the overall IS reachability TLV. The practical maximum is 255 octets minus the 11 octets described above, or 244 octets. Further, there is no defined mechanism for extending the sub-TLV space for a particular neighbor. Thus, wasting sub-TLV space is discouraged.

The metric octets are encoded as a 24-bit unsigned integer. To preclude overflow within an SPF implementation, all metrics greater than or equal to MAX_PATH_METRIC shall be considered to have a metric of MAX_PATH_METRIC. It is easiest to select MAX_PATH_METRIC such that MAX_PATH_METRIC plus a single link metric does not overflow the number of bits for internal metric calculation. We assume that this is 32 bits. Thus, MAX_PATH_METRIC is 4,261,412,864 (0xFE000000, 2^32 - 2^25).

If a link is advertised with the maximum link metric (2^24 - 1), this link should not be considered during the normal SPF computation. This will allow advertisment of a link for other purposes than building the normal Shortest Path Tree. An example is a link that is available for traffic engineering, but not for hop-by-hop routing.

Certain sub-TLVs are proposed here:

Sub-TLV type	Length (octets)	Name

	3	4	Administrative group (color)
	6	4	IPv4 interface address
	8	4	IPv4 neighbor address
	9	4	Maximum link bandwidth
	10	4	Reservable link bandwidth
	11	32	Unreserved bandwidth
	18	3	TE Default metric
	250-254		Reserved for cisco specific extensions
	255		Reserved for future expansion

Each of these sub-TLVs is described below. Unless stated otherwise, multiple occurrences of the information are supported by multiple inclusions of the sub-TLV.

5.1 Sub-TLV 3: Administrative group (color, resource class)

The administrative group sub-TLV contains a 4-octet bit mask assigned by the network administrator. Each set bit corresponds to one administrative group assigned to the interface.

By convention the least significant bit is referred to as 'group 0', and the most significant bit is referred to as 'group 31'.

5.2 Sub-TLV 6: IPv4 interface address

This sub-TLV contains a 4-octet IPv4 address for the interface described by the (main) TLV. This sub-TLV can occur multiple times.

Implementations MUST NOT inject a /32 prefix for the interface address into their routing or forwarding table, because this can lead to forwarding loops when interacting with systems that do not support this sub-TLV.

If a router implements the basic TLV extensions in this document, it is free to add or omit this sub-TLV to the description of an adjacency. If a router implements traffic engineering, it must include this sub-TLV.

5.3 Sub-TLV 8: IPv4 neighbor address

This sub-TLV contains a single IPv4 address for a neighboring router on this link. This sub-TLV can occur multiple times.

Implementations MUST NOT inject a /32 prefix for the neighbor address into their routing or forwarding table, because this can lead to forwarding loops when interacting with systems that do not support this sub-TLV.

If a router implements the basic TLV extensions in this document, it is free to add or omit this sub-TLV to the description of an adjacency. If a router implements traffic engineering, it must include this sub-TLV on point-to-point adjacencies.

5.4 Sub-TLV 9: Maximum link bandwidth

This sub-TLV contains the maximum bandwidth that can be used on this link in this direction (from the system originating the LSP to its neighbors). This is useful for traffic engineering.

The maximum link bandwidth is encoded in 32 bits in IEEE floating point format. The units are bytes (not bits!) per second.

5.5 Sub-TLV 10: Maximum reservable link bandwidth

This sub-TLV contains the maximum amount of bandwidth that can be reserved in this direction on this link. Note that for oversubscription purposes, this can be greater than the bandwidth of the link.

The maximum reservable link bandwidth is encoded in 32 bits in IEEE floating point format. The units are bytes (not bits!) per second.

5.6 Sub-TLV 11: Unreserved bandwidth

This sub-TLV contains the amount of bandwidth reservable on this direction on this link. Note that for oversubscription purposes, this can be greater than the bandwidth of the link.

Because of the need for priority and preemption, each head end

needs

to know the amount of reserved bandwidth at each priority level. Thus, this sub-TLV contains eight 32 bit IEEE floating point numbers.
The units are bytes (not bits!) per second. The values correspond to the bandwidth that can be reserved with a holding of priority 0 through 7, arranged in increasing order with priority 0 occurring at the start of the sub-TLV, and priority 7 at the end of the sub-TLV.

For stability reasons, rapid changes in the values in this sub-TLV should not cause rapid generation of LSPs.

5.7 Sub-TLV 18: Traffic Engineering Default metric

This sub-TLV contains a 24-bit unsigned integer. This metric is administratively assigned and can be used to present a differently weighted topology to traffic engineering SPF calculations.

To preclude overflow within an SPF implementation, all metrics greater than or equal to MAX_PATH_METRIC shall be considered to have a metric of MAX_PATH_METRIC. It is easiest to select MAX_PATH_METRIC such that MAX_PATH_METRIC plus a single link metric does not overflow the number of bits for internal metric calculation. We assume that this is 32 bits. Thus, MAX_PATH_METRIC is 4,261,412,864 (0xFE000000, 2^32 - 2^25).

If a link is advertised without this sub-TLV, traffic engineering SPF calculations must use the normal default metric of this link, which is advertised in the fixed part of TLV 22.

6.0 Security Considerations

This document raises no new security issues for IS-IS.

7.0 Acknowledgments

The authors would like to thank Yakov Rekhter and Dave Katz for their
comments on this work.

8.0 References

[1] draft-ietf-mpls-traffic-eng-00.txt, "Requirements for Traffic Engineering Over MPLS", D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, work in progress.

[2] ISO 10589, "Intermediate System to Intermediate System Intra-Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)" [Also republished as RFC 1142]

[3] RFC 1195, "Use of OSI IS-IS for routing in TCP/IP and dual environments", R.W. Callon, Dec. 1990

9.0 Authors' Addresses

Henk Smit
Cisco Systems, Inc.
210 West Tasman Drive
San Jose, CA 95134
Email: hsmit@cisco.com
Voice: +31 20 342 3736

Tony Li
Juniper Networks, Inc.
385 Ravendale Dr.
Mountain View, CA 94043
Email: tli@juniper.net
Fax: +1 650 526 8001
Voice: +1 650 526 8006

Network Working Group Daniel O. Awduche
Internet Draft UUNET (MCI Worldcom), Inc.
Expiration Date: August 2000
Lou Berger
LabN Consulting, LLC
Der-Hwa Gan
Juniper Networks, Inc.
Tony Li
Procket Networks, Inc.
George Swallow
Cisco Systems, Inc.
Vijay Srinivasan
Torrent Networks, Inc.

February 2000

RSVP-TE: Extensions to RSVP for LSP Tunnels

draft-ietf-mpls-rsvp-lsp-tunnel-05.txt

Status of this Memo

months
   and may be updated, replaced, or obsoleted by other documents at
any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   To view the current status of any Internet-Draft, please check the
   "1id-abstracts.txt" listing contained in an Internet-Drafts Shadow
   Directory, see http://www.ietf.org/shadow.html.

Abstract

   This document describes the use of RSVP, including all the
necessary
   extensions, to establish label-switched paths (LSPs) in MPLS.
Since
   the flow along an LSP is completely identified by the label applied

   at the ingress node of the path, these paths may be treated as
   tunnels. A key application of LSP tunnels is traffic engineering
   with MPLS as specified in [3].

   We propose several additional objects that extend RSVP, allowing
the
   establishment of explicitly routed label switched paths using RSVP
as
   a signaling protocol. The result is the instantiation of label-
   switched tunnels which can be automatically routed away from
network
   failures, congestion, and bottlenecks.

Contents

1. Introduction

Section 2.9 of the MPLS architecture [2] defines a label distribution protocol as a set of procedures by which one Label Switched Router (LSR) informs another of the meaning of labels used to forward traffic between and through them. The MPLS architecture does not assume a single label distribution protocol. This document is a specification of extensions to RSVP for establishing label switched paths (LSPs) in Multi-protocol Label Switching (MPLS) networks.

Several of the new features described in this document were motivated by the requirements for traffic engineering over MPLS (see [3]). In particular, the extended RSVP protocol supports the instantiation of explicitly routed LSPs, with or without resource reservations. It also supports smooth rerouting of LSPs, preemption, and loop detection.

The LSPs created with RSVP can be used to carry the "Traffic Trunks" described in [3]. The LSP which carries a traffic trunk and a traffic trunk are distinct though closely related concepts. For example, two LSPs between the same source and destination could be load shared to carry a single traffic trunk. Conversely several traffic trunks could be carried in the same LSP if, for instance, the LSP were capable of carrying several service classes. The applicability of these extensions is discussed further in [10].

Since the traffic that flows along a label-switched path is defined by the label applied at the ingress node of the LSP, these paths can be treated as tunnels, tunneling below normal IP routing and filtering mechanisms. When an LSP is used in this way we refer to it

as an LSP tunnel.

LSP tunnels allow the implementation of a variety of policies related to network performance optimization. For example, LSP tunnels can be automatically or manually routed away from network failures, congestion, and bottlenecks. Furthermore, multiple parallel LSP tunnels can be established between two nodes, and traffic between the two nodes can be mapped onto the LSP tunnels according to local policy. Although traffic engineering (that is, performance optimization of operational networks) is expected to be an important application of this specification, the extended RSVP protocol can be used in a much wider context.

The purpose of this document is to describe the use of RSVP to establish LSP tunnels. The intent is to fully describe all the objects, packet formats, and procedures required to realize interoperable implementations. A few new objects are also defined that enhance management and diagnostics of LSP tunnels.

The document also describes a means of rapid node failure detection via a new HELLO message.

All objects and messages described in this specification are optional with respect to RSVP. This document discusses what happens when an object described here is not supported by a node.

Throughout this document, the discussion will be restricted to unicast label switched paths. Multicast LSPs are left for further study.

1.1. Background

Hosts and routers that support both RSVP [1] and Multi-Protocol

Label
   Switching [2] can associate labels with RSVP flows. When MPLS and
   RSVP are combined, the definition of a flow can be made more
   flexible. Once a label switched path (LSP) is established, the
   traffic through the path is defined by the label applied at the
   ingress node of the LSP. The mapping of label to traffic can be
   accomplished using a number of different criteria. The set of
   packets that are assigned the same label value by a specific node
are
   said to belong to the same forwarding equivalence class (FEC) (see
   [2]), and effectively define the "RSVP flow." When traffic is
mapped
   onto a label-switched path in this way, we call the LSP an "LSP
   Tunnel". When labels are associated with traffic flows, it becomes
   possible for a router to identify the appropriate reservation state
   for a packet based on the packet's label value.

   The signaling protocol model uses downstream-on-demand label
   distribution. A request to bind labels to a specific LSP tunnel is
   initiated by an ingress node through the RSVP Path message. For
this
   purpose, the RSVP Path message is augmented with a LABEL_REQUEST
   object. Labels are allocated downstream and distributed (propagated
   upstream) by means of the RSVP Resv message. For this purpose, the
   RSVP Resv message is extended with a special LABEL object. Label
   stacking is also supported. The procedures for label allocation,
   distribution, binding, and stacking are described in subsequent
   sections of this document.

   The signaling protocol model also supports explicit routing
   capability. This is accomplished by incorporating a simple
   EXPLICIT_ROUTE object into RSVP Path messages. The EXPLICIT_ROUTE
   object encapsulates a concatenation of hops which constitutes the
   explicitly routed path. Using this object, the paths taken by
label-
   switched RSVP-MPLS flows can be pre-determined, independent of
   conventional IP routing. The explicitly routed path can be
   administratively specified, or automatically computed by a suitable

   entity based on QoS and policy requirements, taking into

consideration the prevailing network state. In general, path computation can be control-driven or data-driven. The mechanisms, processes, and algorithms used to compute explicitly routed paths are beyond the scope of this specification.

One useful application of explicit routing is traffic engineering. Using explicitly routed LSPs, a node at the ingress edge of an MPLS domain can control the path through which traffic traverses from itself, through the MPLS network, to an egress node. Explicit routing can be used to optimize the utilization of network resources and enhance traffic oriented performance characteristics.

The concept of explicitly routed label switched paths can be generalized through the notion of abstract nodes. An abstract node is a group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node. Using this concept of abstraction, an explicitly routed LSP can be specified as a sequence of IP prefixes or a sequence of Autonomous Systems.

The signaling protocol model supports the specification of an explicit path as a sequence of strict and loose routes. The combination of abstract nodes, and strict and loose routes significantly enhances the flexibility of path definitions.

An advantage of using RSVP to establish LSP tunnels is that it enables the allocation of resources along the path. For example, bandwidth can be allocated to an LSP tunnel using standard RSVP reservations and Integrated Services service classes [4].

While resource reservations are useful, they are not mandatory. Indeed, an LSP can be instantiated without any resource reservations whatsoever. Such LSPs without resource reservations can be used, for example, to carry best effort traffic. They can also be used in many other contexts, including implementation of fall-back and recovery policies under fault conditions, and so forth.

1.2. 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 [6].

The reader is assumed to be familiar with the terminology in [1], [2]
and [3].

Abstract Node

A group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node.

Explicitly Routed LSP

An LSP whose path is established by a means other than normal IP routing.

Label Switched Path

The path created by the concatenation of one or more label switched hops, allowing a packet to be forwarded by swapping labels from an MPLS node to another MPLS node. For a more precise definition see [2].

LSP

A Label Switched Path

LSP Tunnel

An LSP which is used to tunnel below normal IP routing and/or filtering mechanisms.

Traffic Trunk

An set of flows aggregated by their service class and then placed on an LSP or set of LSPs. For further discussion see

[3].

2. Overview

2.1. LSP Tunnels

According to [1], "RSVP defines a 'session' to be a data flow with a particular destination and transport-layer protocol." However, when RSVP and MPLS are combined, a flow or session can be defined with greater flexibility and generality. The ingress node of an LSP can use a variety of means to determine which packets are assigned a particular label. Once a label is assigned to a set of packets, the label effectively defines the "flow" through the LSP. We refer to such an LSP as an "LSP tunnel" because the traffic through it is

opaque to intermediate nodes along the label switched path.

New RSVP SESSION objects, called LSP_TUNNEL_IPv4 and LSP_TUNNEL_IPv6 have been defined to support the LSP tunnel feature. The semantics of these objects, from the perspective of a node along the label switched path, is that traffic belonging to the LSP tunnel is identified solely on the basis of packets arriving from the PHOP or "previous hop" (see [1]) with the particular label value(s) assigned by this node to upstream senders to the session. In fact, the IPv4(v6) that appears in the object name only denotes that the destination address is an IPv4(v6) address. When we refer to these objects generically, we use the term LSP_TUNNEL Session.

2.2. Operation of LSP Tunnels

This section summarizes some of the features supported by RSVP as extended by this document related to the operation of LSP tunnels. These include: (1) the capability to establish LSP tunnels with or

without QoS requirements, (2) the capability to dynamically reroute an established LSP tunnel, (3) the capability to observe the actual route traversed by an established LSP tunnel, (4) the capability to identify and diagnose LSP tunnels, (5) the capability to preempt an established LSP tunnel under administrative policy control, and (6) the capability to perform downstream-on-demand label allocation, distribution, and binding. In the following paragraphs, these features are briefly described. More detailed descriptions can be found in subsequent sections of this document.

To create an LSP tunnel, the first MPLS node on the path -- that is, the sender node with respect to the path -- creates an RSVP Path message with a session type of LSP_TUNNEL_IPv4 or LSP_TUNNEL_IPv6 and inserts a LABEL_REQUEST object into the Path message. The LABEL_REQUEST object indicates that a label binding for this path is requested and also provides an indication of the network layer protocol that is to be carried over this path. The reason for this is that the network layer protocol sent down an LSP cannot be assumed to be IP and cannot be deduced from the L2 header, which simply identifies the higher layer protocol as MPLS.

If the sender node has knowledge of a route that has high likelihood of meeting the tunnel's QoS requirements, or that makes efficient use of network resources, or that satisfies some policy criteria, the node can decide to use the route for some or all of its sessions. To do this, the sender node adds an EXPLICIT_ROUTE object to the RSVP Path message. The EXPLICIT_ROUTE object specifies the route as a sequence of abstract nodes.

If, after a session has been successfully established and the sender

node discovers a better route, the sender can dynamically reroute the session by simply changing the EXPLICIT_ROUTE object. If problems are encountered with an EXPLICIT_ROUTE object, either because it causes a routing loop or because some intermediate routers do not support it, the sender node is notified.

By adding a RECORD_ROUTE object to the Path message, the sender node can receive information about the actual route that the LSP tunnel traverses. The sender node can also use this object to request notification from the network concerning changes to the routing path.
The RECORD_ROUTE object is analogous to a path vector, and hence can be used for loop detection.

Finally, a SESSION_ATTRIBUTE object can be added to Path messages to aid in session identification and diagnostics. Additional control information, such as preemption, priority, and local-protection, are also included in this object.

When the EXPLICIT_ROUTE object (ERO) is present, the Path message is forwarded towards its destination along a path specified by the ERO.
Each node along the path records the ERO in its path state block. Nodes may also modify the ERO before forwarding the Path message. In this case the modified ERO SHOULD be stored in the path state block in addition to the received ERO.

The LABEL_REQUEST object requests intermediate routers and receiver nodes to provide a label binding for the session. If a node is incapable of providing a label binding, it sends a PathErr message with an "unknown object class" error. If the LABEL_REQUEST object is not supported end to end, the sender node will be notified by the first node which does not provide this support.

The destination node of a label-switched path responds to a LABEL_REQUEST by including a LABEL object in its response RSVP Resv message. The LABEL object is inserted in the filter spec list immediately following the filter spec to which it pertains.

The Resv message is sent back upstream towards the sender, following the path state created by the Path message, in reverse order. Note

that if the path state was created by use of an ERO, then the Resv message will follow the reverse path of the ERO.

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, 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. The label sent upstream in the LABEL

object is the label which this node will use to identify incoming traffic associated with this LSP tunnel. This label also serves as shorthand for the Filter Spec. The node can now update its "Incoming Label Map" (ILM), which is used to map incoming labeled packets to a "Next Hop Label Forwarding Entry" (NHLFE), see [2].

When the Resv message propagates upstream to the sender node, a label-switched path is effectively established.

2.3. Service Classes

This document does not restrict the type of Integrated Service requests for reservations. However, an implementation should support the Controlled-Load service [4] and the Class-of-Service service, see Section 4.8.

2.4. Reservation Styles

The receiver node can select from among a set of possible reservation styles for each session, and each RSVP session must have a

particular
style. Senders have no influence on the choice of reservation
style.
The receiver can choose different reservation styles for different
LSPs.

An RSVP session can result in one or more LSPs, depending on the
reservation style chosen.

Some reservation styles, such as FF, dedicate a particular
reservation to an individual sender node. Other reservation
styles,
such as WF and SE, can share a reservation among several sender
nodes. The following sections discuss the different reservation
styles and their advantages and disadvantages. A more detailed
discussion of reservation styles can be found in [1].

2.4.1. Fixed Filter (FF) Style

The Fixed Filter (FF) reservation style creates a distinct
reservation for traffic from each sender that is not shared by
other
senders. This style is common for applications in which traffic
from
each sender is likely to be concurrent and independent. The total
amount of reserved bandwidth on a link for sessions using FF is the
sum of the reservations for the individual senders.

Because each sender has its own reservation, a unique label is
assigned to each sender. This can result in a point-to-point LSP

between every sender/receiver pair.

2.4.2. Wildcard Filter (WF) Style

With the Wildcard Filter (WF) reservation style, a single shared
reservation is used for all senders to a session. The total
reservation on a link remains the same regardless of the number of

senders.

A single multipoint-to-point label-switched-path is created for all senders to the session. On links that senders to the session share, a single label value is allocated to the session. If there is only one sender, the LSP looks like a normal point-to-point connection. When multiple senders are present, a multipoint-to-point LSP (a reversed tree) is created.

This style is useful for applications in which not all senders send traffic at the same time. A phone conference, for example, is an application where not all speakers talk at the same time. If, however, all senders send simultaneously, then there is no means of getting the proper reservations made. Either the reserved bandwidth on links close to the destination will be less than what is required or then the reserved bandwidth on links close to some senders will be greater than what is required. This restricts the applicability of WF for traffic engineering purposes.

Furthermore, because of the merging rules of WF, EXPLICIT_ROUTE objects cannot be used with WF reservations. As a result of this issue and the lack of applicability to traffic engineering, use of WF is not considered in this document.

2.4.3. Shared Explicit (SE) Style

The Shared Explicit (SE) style allows a receiver to explicitly specify the senders to be included in a reservation. There is a single reservation on a link for all the senders listed. Because each sender is explicitly listed in the Resv message, different labels may be assigned to different senders, thereby creating separate LSPs.

SE style reservations can be provided using multipoint-to-point label-switched-path or LSP per sender. Multipoint-to-point LSPs may be used when path messages do not carry the EXPLICIT_ROUTE object, or when Path messages have identical EXPLICIT_ROUTE objects. In either of these cases a common label may be assigned.

Path messages from different senders can each carry their own ERO, and the paths taken by the senders can converge and diverge at any point in the network topology. When Path messages have differing EXPLICIT_ROUTE objects, separate LSPs for each EXPLICIT_ROUTE object must be established.

2.5. Rerouting LSP Tunnels

One of the requirements for Traffic Engineering is the capability to reroute an established LSP tunnel under a number of conditions, based on administrative policy. For example, in some contexts, an administrative policy may dictate that a given LSP tunnel is to be rerouted when a more "optimal" route becomes available. Another important context when LSP tunnel reroute is usually required is upon failure of a resource along the tunnel's established path. Under some policies, it may also be necessary to return the LSP tunnel to its original path when the failed resource becomes re-activated.

In general, it is highly desirable not to disrupt traffic, or adversely impact network operations while LSP tunnel rerouting is in progress. This adaptive and smooth rerouting requirement necessitates establishing a new LSP tunnel and transferring traffic from the old LSP tunnel onto it before tearing down the old LSP tunnel. This concept is called "make-before-break." A problem can arise because the old and new LSP tunnels might compete with other for resources on network segments which they have in common. Depending on availability of resources, this competition can cause Admission Control to prevent the new tunnel from being established. An advantage of using RSVP to establish LSP tunnels is that it solves this problem very elegantly.

To support make-before-break in a smooth fashion, it is necessary

that on links that are common to the old and new LSPs, resources used by the old LSP tunnel should not be released before traffic is transitioned to the new LSP tunnel, and reservations should not be counted twice because this might cause Admission Control to reject the new LSP tunnel.

The combination of the LSP_TUNNEL SESSION object and the SE reservation style naturally achieves smooth transitions. The basic idea is that the old and new LSP tunnels share resources along links which they have in common. The LSP_TUNNEL SESSION object is used to narrow the scope of the RSVP session to the particular tunnel in question. To uniquely identify a tunnel, we use the combination of the destination IP address (an address of the node which is the egress of the tunnel), a Tunnel ID, and the tunnel ingress node's IP address, which is placed in the Extended Tunnel ID field.

During the reroute operation, the tunnel ingress needs to appear as two different senders to the RSVP session. This is achieved by the inclusion of an "LSP ID", which is carried in the SENDER_TEMPLATE and FILTER_SPEC objects. Since the semantics of these objects are changed, a new C-Type is assigned.

To effect a reroute, the ingress node picks a new LSP ID and forms a new SENDER_TEMPLATE. The ingress node then creates a new ERO to define the new path. Thereafter the node sends a new Path Message using the original SESSION object and the new SENDER_TEMPLATE and ERO. It continues to use the old LSP and refresh the old Path message. On links that are not held in common, the new Path message is treated as a conventional new LSP tunnel setup. On links held in common, the shared SESSION object and SE style allow the LSP to be established sharing resources with the old LSP. Once the ingress node receives a Resv message for the new LSP, it can transition traffic to it and tear down the old LSP.

3. LSP Tunnel related Message Formats

Five new objects are defined in this section:

Object name	Applicable RSVP messages
LABEL_REQUEST	Path
LABEL	Resv
EXPLICIT_ROUTE	Path
RECORD_ROUTE	Path, Resv
SESSION_ATTRIBUTE	Path

New C-Types are also assigned for the SESSION, SENDER_TEMPLATE, FILTER_SPEC, FLOWSPEC objects.

Detailed descriptions of the new objects are given in later sections.

All new objects are OPTIONAL with respect to RSVP. An implementation can choose to support a subset of objects. However, the LABEL_REQUEST and LABEL objects are mandatory with respect to this specification.

The LABEL and RECORD_ROUTE objects, are sender specific. In Resv messages they MUST appear after the associated FILTER_SPEC and prior to any subsequent FILTER_SPEC.

The relative placement of EXPLICIT_ROUTE, LABEL_REQUEST, and SESSION_ATTRIBUTE objects is simply a recommendation. The ordering of these objects is not important, so an implementation MUST be

prepared to accept objects in any order.

3.1. Path Message

The format of the Path message is as follows:

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

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

3.2. Resv Message

The format of the Resv message is as follows:

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

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

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

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

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

Note: LABEL and RECORD_ROUTE (if present), are bound to the preceding FILTER_SPEC. No more than one LABEL and/or RECORD_ROUTE may follow each FILTER_SPEC.

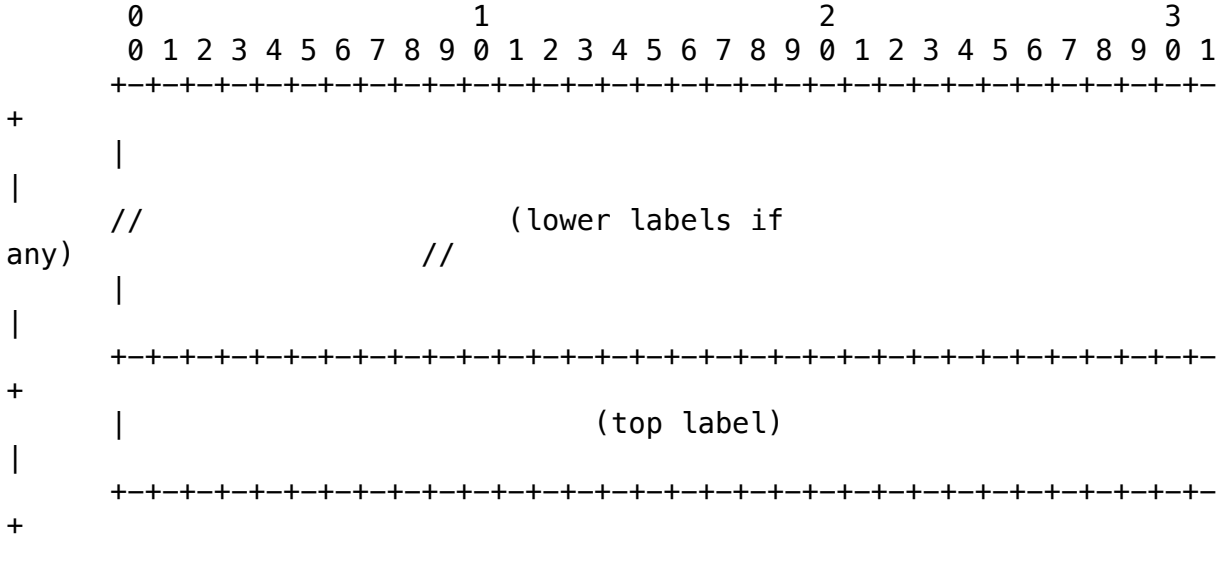
4. LSP Tunnel related Objects

4.1. Label Object

Labels MAY be carried in Resv messages. For the FF and SE styles, a label is associated with each sender. The label for a sender MUST immediately follow the FILTER_SPEC for that sender in the Resv message.

The LABEL object has the following format:

LABEL class = 16, C_Type = 1



The contents of a LABEL object are a stack of labels, where each label is encoded in 4 octets. The top of the stack is in the right 4 octets of the object contents. A LABEL object that contains no labels is illegal.

Each generic MPLS label is an unsigned integer in the range 0 through 1048575. Generic MPLS labels and FR labels are encoded right aligned in 4 octets. ATM labels are encoded with the VPI right justified in bits 0-15 and the VCI right justified in bits 16-31.

The decision concerning whether to create a label stack with more than one label, when to push a new label, and when to pop the label

stack is not addressed in this document. For implementations that do not support a label stack, only the top label is examined. The rest of the label stack SHOULD be passed through unchanged. Such implementations are REQUIRED to generate a label stack of depth 1 when initiating the first LABEL.

4.1.1. Handling Label Objects in Resv messages

A router uses the top label carried in the LABEL object as the outgoing label associated with the sender. The router allocates a new label and binds it to the incoming interface of this session/sender. This is the same interface that the router uses to forward Resv messages to the previous hops.

In MPLS a node may support multiple label spaces, perhaps associating a unique space with each incoming interface. For the purposes of the following discussion, the term "same label" means the identical label

value drawn from the identical label space. Further, the following applies only to unicast sessions.

If a node receives a Resv message that has assigned the same label value to multiple senders, then that node MAY also assign the same value to those same senders or to any subset of those senders. Note that if a node intends to police individual senders to a session, it MUST assign unique labels to those senders.

Labels received in Resv messages on different interfaces are always considered to be different even if the label value is the same.

To construct a new LABEL object, the router replaces the top label (from the received Resv message) with the locally allocated new label. The router then sends the new LABEL object as part of the Resv message to the previous hop. The LABEL object SHOULD be kept in the Reservation State Block. It is then used in the next Resv refresh event for formatting the Resv message.

A router is expected to send a Resv message before its refresh timers expire if the contents of the LABEL object change.

4.1.2. Non-support of the Label Object

Under normal circumstances, a node should never receive a LABEL object in a Resv message unless it had included a LABEL_REQUEST object in the corresponding Path message. However, an RSVP router that does not recognize the LABEL object sends a ResvErr with the error code "Unknown object class" toward the receiver. This causes

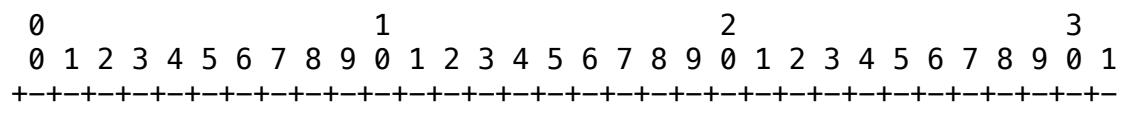
the reservation to fail.

4.2. Label Request Object

The Label Request Class is 19. Currently there three possible C_Types. Type 1 is a Label Request without label range. Type 2 is a label request with an ATM label range. Type 3 is a label request with a Frame Relay label range. The LABEL_REQUEST object formats are shown below.

Label Request without Label Range

Class = 19, C_Type = 1



Reserved

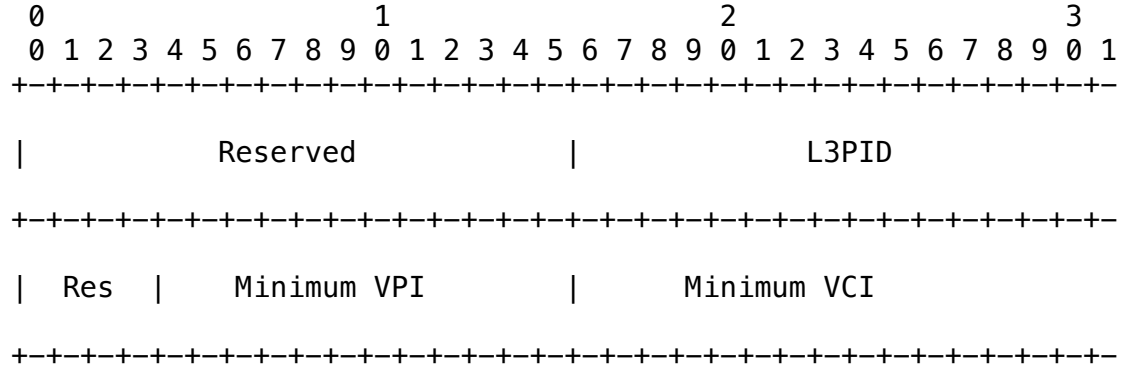
This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

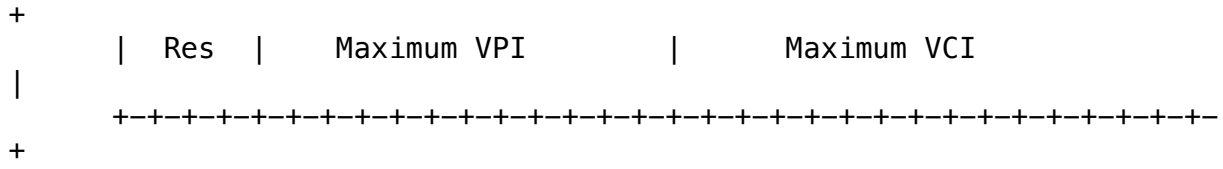
L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

Label Request with ATM Label Range

Class = 19, C_Type = 2





Reserved (Res)

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

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

Minimum VPI (12 bits)

This 12 bit field specifies the lower bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Minimum VCI (16 bits)

This 16 bit field specifies the lower bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VPI (12 bits)

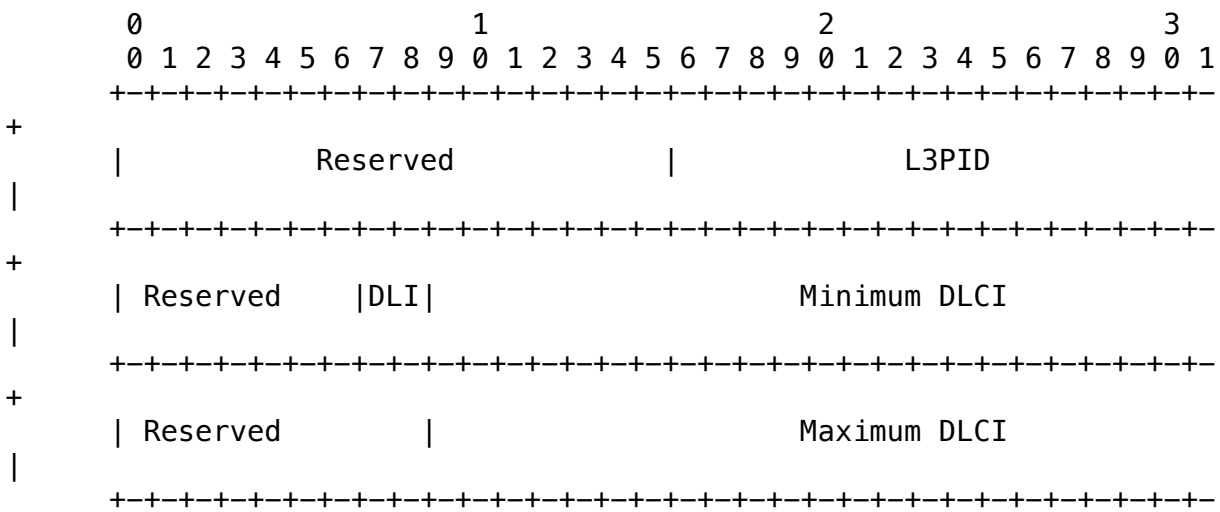
This 12 bit field specifies the upper bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VCI (16 bits)

This 16 bit field specifies the upper bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Label Request with Frame Relay Label Range

Class = 19, C_Type = 3



+

Reserved

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

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

DLI

DLCI Length Indicator. The number of bits in the DLCI. The following values are supported:

Len	DLCI bits
0	10
1	17
2	23

Minimum DLCI

This 23-bit field specifies the lower bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

Maximum DLCI

This 23-bit field specifies the upper bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

4.2.1. Handling of LABEL_REQUEST

To establish an LSP tunnel the sender creates a Path message with a LABEL_REQUEST object. The LABEL_REQUEST object indicates that a label binding for this path is requested and provides an indication of the network layer protocol that is to be carried over this path. This permits non-IP network layer protocols to be sent down an LSP. This information can also be useful in actual label allocation, because some reserved labels are protocol specific, see [5].

The LABEL_REQUEST SHOULD be stored in the Path State Block, so that Path refresh messages will also contain the LABEL_REQUEST object. When the Path message reaches the receiver, the presence of the LABEL_REQUEST object triggers the receiver to allocate a label and to place the label in the LABEL object for the corresponding Resv message. If a label range was specified, the label MUST be allocated from that range. A receiver that accepts a LABEL_REQUEST object MUST include a LABEL object in Resv messages pertaining to that Path message. If a LABEL_REQUEST object was not present in the Path message, a node MUST NOT include a LABEL object in a Resv message for that Path message's session and PHOP.

A node that sends a LABEL_REQUEST object MUST be ready to accept and correctly process a LABEL object in the corresponding Resv messages.

A node that recognizes a LABEL_REQUEST object, but that is unable to support it (possibly because of a failure to allocate labels) SHOULD send a PathErr with the error code "Routing problem" and the error value "MPLS label allocation failure." This includes the case where a label range has been specified and a label cannot be allocated from that range.

If the receiver cannot support the protocol L3PID, it SHOULD send a PathErr with the error code "Routing problem" and the error value "Unsupported L3PID." This causes the RSVP session to fail.

4.2.2. Non-support of the Label Request Object

An RSVP router that does not recognize the LABEL_REQUEST object sends a PathErr with the error code "Unknown object class" toward the sender. An RSVP router that recognizes the LABEL_REQUEST object but does not recognize the C_Type sends a PathErr with the error code "Unknown object C_Type" toward the sender. This causes the path setup to fail. The sender should notify management that a LSP cannot be established and possibly take action to continue the reservation without the LABEL_REQUEST.

RSVP is designed to cope gracefully with non-RSVP routers anywhere between senders and receivers. However, obviously, non-RSVP routers cannot convey labels via RSVP. This means that if a router has a neighbor that is known to not be RSVP capable, the router MUST NOT advertise the LABEL_REQUEST object when sending messages that pass through the non-RSVP routers. The router SHOULD send a PathErr back to the sender, with the error code "Routing problem" and the error value "MPLS being negotiated, but a non-RSVP capable router stands in the path." This same message SHOULD be sent, if a router receives a LABEL_REQUEST object in a message from a non-RSVP capable router. See [1] for a description of how a downstream router can determine the presence of non-RSVP routers.

4.3. Explicit Route Object

Explicit routes are specified via the EXPLICIT_ROUTE object (ERO). The Explicit Route Class is 20. Currently one C_Type is defined, Type 1 Explicit Route. The EXPLICIT_ROUTE object has the following format:

```
      Class = 20, C_Type = 1

       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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
+
      |                                                               
|
      //                                                              
(Subobjects)                          //
      |                                                               
|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
+
```

Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. The subobjects are defined in section 4.3.3 below.

If a Path message contains multiple EXPLICIT_ROUTE objects, only the first object is meaningful. Subsequent EXPLICIT_ROUTE objects MAY be ignored and SHOULD NOT be propagated.

4.3.1. Applicability

The EXPLICIT_ROUTE object is intended to be used only for unicast situations. Applications of explicit routing to multicast are a topic for further research.

The EXPLICIT_ROUTE object is to be used only when all routers along the explicit route support RSVP and the EXPLICIT_ROUTE object. The

EXPLICIT_ROUTE object is assigned a class value of the form 0bbbbbbb.
RSVP routers that do not support the object will therefore respond with an "Unknown Object Class" error.

4.3.2. Semantics of the Explicit Route Object

An explicit route is a particular path in the network topology. Typically, the explicit route is determined by a node, with the intent of directing traffic along that path.

An explicit route is described as a list of groups of nodes along the explicit route. Certain operations to be performed along the path can also be encoded in the EXPLICIT_ROUTE object.

In addition to the ability to identify specific nodes along the path, an explicit route can identify a group of nodes that must be traversed along the path. This capability allows the routing system a significant amount of local flexibility in fulfilling a request for an explicit route. This capability allows the generator of the explicit route to have imperfect information about the details of the path.

The explicit route is encoded as a series of subobjects contained in an EXPLICIT_ROUTE object. Each subobject may identify a group of nodes in the explicit route or may specify an operation to be performed along the path. An explicit route then becomes a specification of groups of nodes to be traversed and a set of operations to be performed along the path.

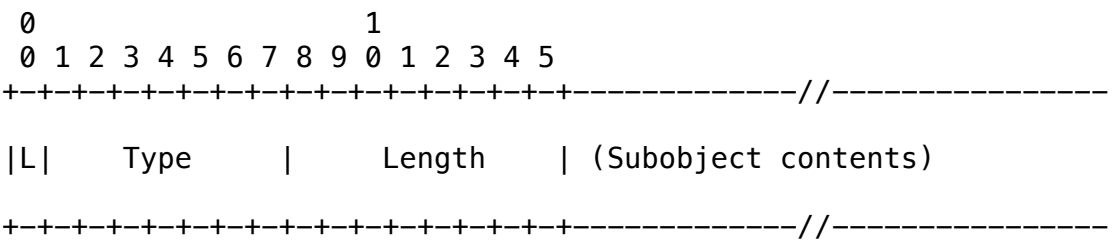
To formalize the discussion, we call each group of nodes an abstract node. Thus, we say that an explicit route is a specification of a set of abstract nodes to be traversed and a set operations to be performed along that path. If an abstract node consists of only one node, we refer to it as a simple abstract node.

As an example of the concept of abstract nodes, consider an explicit route that consists solely of Autonomous System number subobjects. Each subobject corresponds to an Autonomous System in the global topology. In this case, each Autonomous System is an abstract node,

and the explicit route is a path that includes each of the specified
Autonomous Systems. There may be multiple hops within each Autonomous System, but these are opaque to the source node for the explicit route.

4.3.3. Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. Each subobject has the form:



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

The Type indicates the type of contents of the subobject. Currently defined values are:

0	Reserved
1	IPv4 prefix
2	IPv6 prefix
32	Autonomous system number

Length

The Length contains the total length of the subobject in bytes, including the L, Type and Length fields. The Length MUST be at least 4, and MUST be a multiple of 4.

4.3.3.1. Strict and Loose Subobjects

The L bit in the subobject is a one-bit attribute. If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.' For brevity, we say that if the value of the subobject attribute is 'loose' then it is a 'loose subobject.' Otherwise, it's a 'strict subobject.' Further, we say that the abstract node of a strict or loose subobject is a strict or a loose node, respectively. Loose and strict nodes are always interpreted relative to their prior abstract nodes.

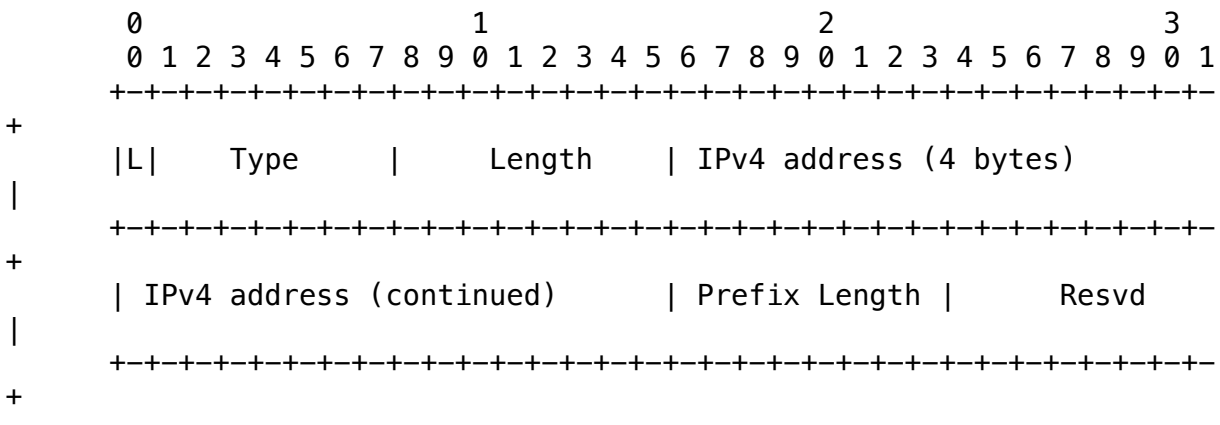
The path between a strict node and its preceding node MUST include

only network nodes from the strict node and its preceding abstract node.

The path between a loose node and its preceding node MAY include other network nodes that are not part of the strict node or its preceding abstract node.

The L bit has no meaning in operation subobjects.

4.3.3.2. Subobject 1: IPv4 prefix



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

An IPv4 address. This address is treated as a prefix based on the prefix length value below. Bits beyond the prefix are ignored on receipt and SHOULD be set to zero on transmission.

Prefix length

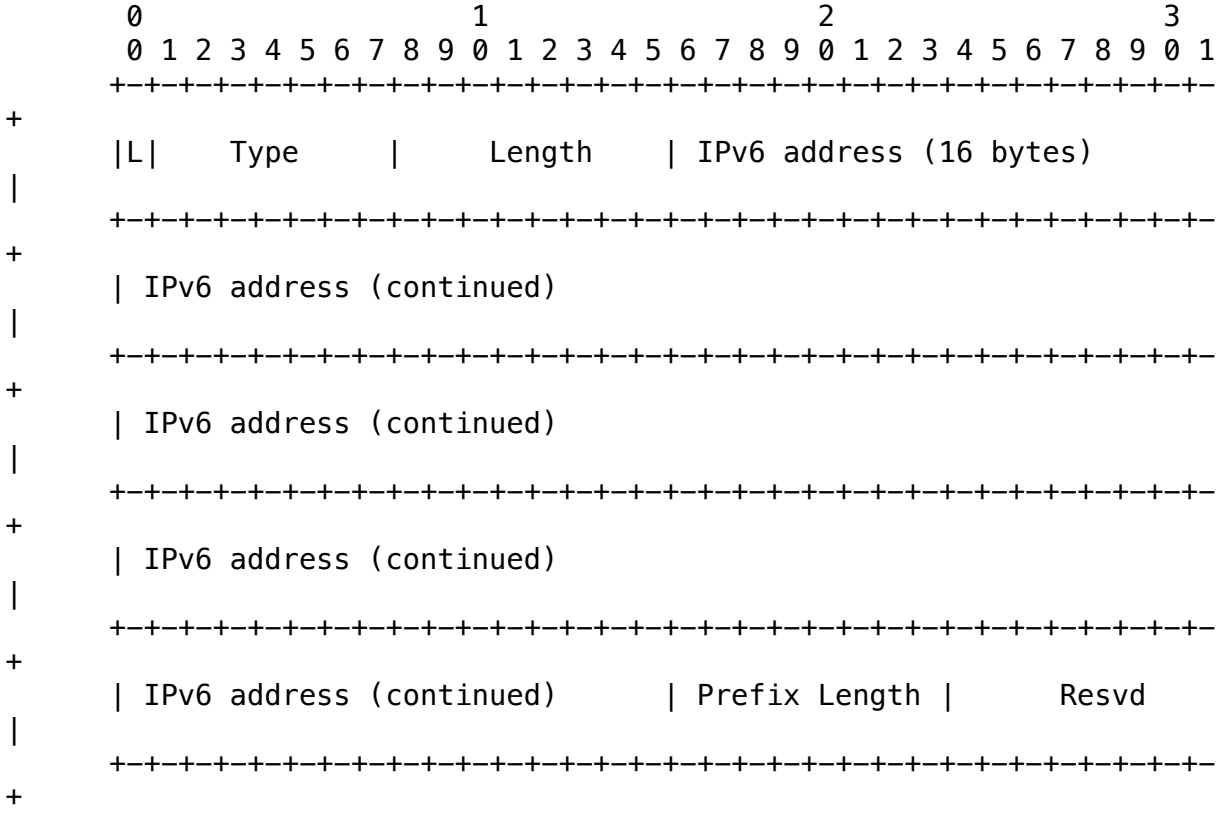
Length in bits of the IPv4 prefix

Padding

Zero on transmission. Ignored on receipt.

The contents of an IPv4 prefix subobject are a 4-octet IPv4 address, a 1-octet prefix length, and a 1-octet pad. The abstract node represented by this subobject is the set of nodes that have an IP address which lies within this prefix. Note that a prefix length of 32 indicates a single IPv4 node.

4.3.3.3. Subobject 2: IPv6 Prefix



L

The L bit is an attribute of the subobject. The L bit is

set
         if the subobject represents a loose hop in the explicit
route.
         If the bit is not set, the subobject represents a strict hop
in
         the explicit route.

      Type

         0x02 IPv6 address

      Length

         The Length contains the total length of the subobject in
         bytes, including the Type and Length fields. The Length
         is always 20.

      IPv6 address

         An IPv6 address. This address is treated as a prefix based
on
         the prefix length value below. Bits beyond the prefix are
         ignored on receipt and SHOULD be set to zero on transmission.

       Prefix Length

          Length in bits of the IPv6 prefix.

       Padding

          Zero on transmission. Ignored on receipt.

   The contents of an IPv6 prefix subobject are a 16-octet IPv6

address,
   a 1-octet prefix length, and a 1-octet pad. The abstract node
   represented by this subobject is the set of nodes that have an IP
   address which lies within this prefix. Note that a prefix length
of
   128 indicates a single IPv6 node.

4.3.3.4. Subobject 32: Autonomous System Number

   The contents of an Autonomous System (AS) number subobject are a 2-
   octet AS number. The abstract node represented by this subobject
is
   the set of nodes belonging to the autonomous system.

   The length of the AS number subobject is 4 octets.

4.3.4. Processing of the Explicit Route Object

4.3.4.1. Selection of the Next Hop

   A node receiving a Path message containing an EXPLICIT_ROUTE object
   must determine the next hop for this path. This is necessary
because
   the next abstract node along the explicit route might be an IP
subnet
   or an Autonomous System. Therefore, selection of this next hop may
   involve a decision from a set of feasible alternatives. The
criteria
   used to make a selection from feasible alternatives is
implementation
   dependent and can also be impacted by local policy, and is beyond
the
   scope of this specification. However, it is assumed that each node
   will make a best effort attempt to determine a loop-free path.
Note
   that paths so determined can be overridden by local policy.

   To determine the next hop for the path, a node performs the
following

steps:

1) The node receiving the RSVP message MUST first evaluate the first subobject. If the node is not part of the abstract node described by the first subobject, it has received the message in error and SHOULD return a "Bad initial subobject" error. If the first subobject is an operation subobject, the message is in error and the system SHOULD return a "Bad EXPLICIT_ROUTE object" error. If there is no first subobject, the message is also in error and the system SHOULD return a "Bad EXPLICIT_ROUTE object" error.

2) If there is no second subobject, this indicates the end of the explicit route. The EXPLICIT_ROUTE object SHOULD be removed from the Path message. This node may or may not be the end of the path. Processing continues with section 4.3.4.2, where a new EXPLICIT_ROUTE object MAY be added to the Path message.

3) Next, the node evaluates the second subobject. If the subobject is an operation subobject, the node pops the subobject from the EXPLICIT ROUTE object , records the subobject, and continues processing with step 2, above. Note that this changes the third subobject into the second subobject (hence "pop") in subsequent processing. The precise operations to be performed by this node must be defined by the operation subobject.

4) If the node is also a part of the abstract node described by the second subobject, then the node deletes the first subobject and continues processing with step 2, above. Note that this makes the second subobject into the first subobject of the next iteration and allows the node to identify the next abstract node on the path of the message after possible repeated application(s) of steps 2-4.

5) Abstract Node Border Case: The node determines whether it is topologically adjacent to the abstract node described by the second subobject. If so, the node selects a particular next hop which is a member of the abstract node. The node then deletes the first subobject and continues processing with section 4.3.4.2.

6) Interior of the Abstract Node Case: Otherwise, the node selects a
next hop within the abstract node of the first subobject (which the
node belongs to) that is along the path to the abstract node of the
second subobject (which is the next abstract node). If no such path
exists then there are two cases:

6a) If the second subobject is a strict subobject, there is an error
and the node SHOULD return a "Bad strict node" error.

6b) Otherwise, if the second subobject is a loose subobject, the node

selects any next hop that is along the path to the next abstract
node. If no path exists, there is an error, and the node SHOULD
return a "Bad loose node" error.

7) Finally, the node replaces the first subobject with any subobject
that denotes an abstract node containing the next hop. This is
necessary so that when the explicit route is received by the next
hop, it will be accepted.

4.3.4.2. Adding subobjects to the Explicit Route Object

After selecting a next hop, the node MAY alter the explicit route in
the following ways.

If, as part of executing the algorithm in section 4.3.4.1, the
EXPLICIT_ROUTE object is removed, the node MAY add a new
EXPLICIT_ROUTE object.

Otherwise, if the node is a member of the abstract node for the first
subobject, a series of subobjects MAY be inserted before the first
subobject or MAY replace the first subobject. Each subobject in

this
series MUST denote an abstract node that is a subset of the current abstract node.

Alternately, if the first subobject is a loose subobject, an arbitrary series of subobjects MAY be inserted prior to the first subobject.

4.3.5. Loops

While the EXPLICIT_ROUTE object is of finite length, the existence of loose nodes implies that it is possible to construct forwarding loops during transients in the underlying routing protocol. This can be detected by the originator of the explicit route through the use of another opaque route object called the RECORD_ROUTE object. The RECORD_ROUTE object is used to collect detailed path information and is useful for loop detection and for diagnostics.

4.3.6. Non-support of the Explicit Route Object

An RSVP router that does not recognize the EXPLICIT_ROUTE object sends a PathErr with the error code "Unknown object class" toward the sender. This causes the path setup to fail. The sender should notify management that a LSP cannot be established and possibly take action to continue the reservation without the EXPLICIT_ROUTE or via

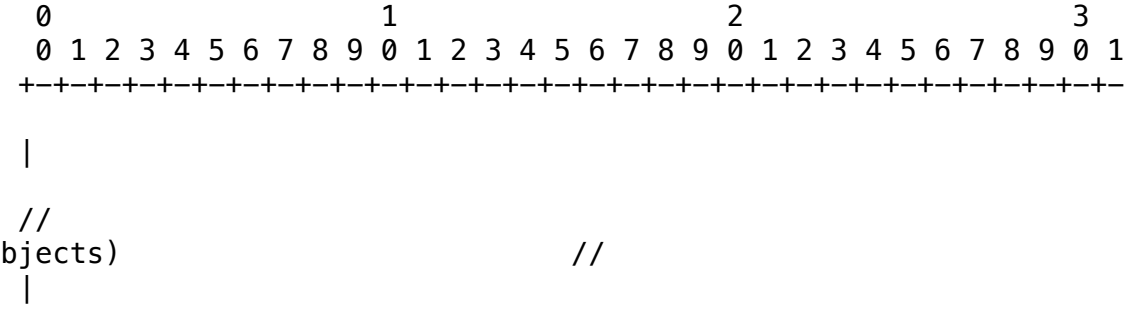
a different explicit route.

4.4. Record Route Object

Routes can be recorded via the RECORD_ROUTE object (RRO). The

Record
   Route Class is 21. Currently one C_Type is defined, Type 1
Record
   Route. The RECORD_ROUTE object has the following format:

      Class = 21, C_Type = 1



      Subobjects

         The contents of a RECORD_ROUTE object are a series
of
         variable-length data items called subobjects. The
subobjects
         are defined in section 4.4.1 below.

   The RRO can be present in both RSVP Path and Resv messages. If a
   Path message contains multiple RROs, only the first RRO is
   meaningful. Subsequent RROs SHOULD be ignored and SHOULD NOT be
   propagated. Similarly, if in a Resv message multiple RROs are
   encountered following a FILTER_SPEC before another FILTER_SPEC is
   encountered, only the first RRO is meaningful. Subsequent RROs
   SHOULD be ignored and SHOULD NOT be propagated.

4.4.1. Subobjects

   The contents of a RECORD_ROUTE object are a series of variable-
length
   data items called subobjects. Each subobject has its own Length
   field. The length contains the total length of the subobject in
   bytes, including the Type and Length fields. The length MUST
always
   be a multiple of 4, and at least 4.

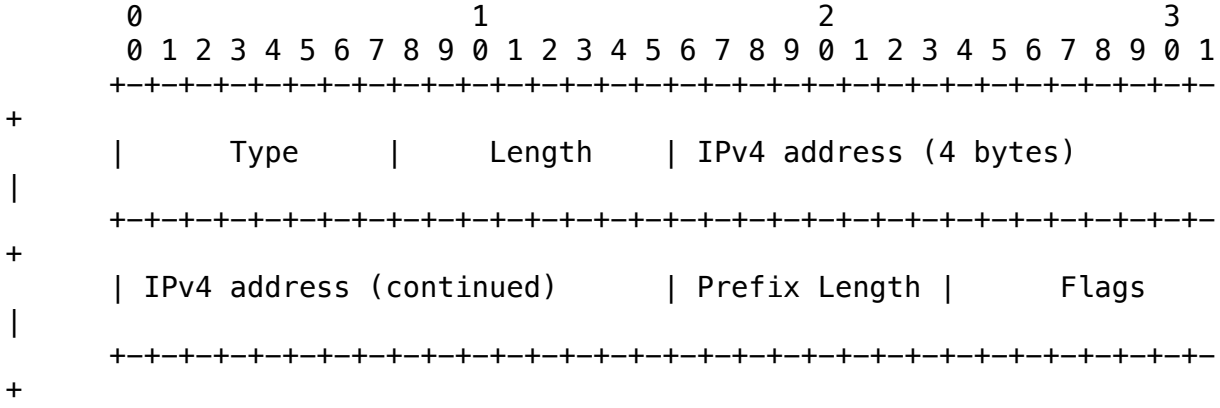
   Subobjects are organized as a last-in-first-out stack. The first

subobject relative to the beginning of RRO is considered the top. The last subobject is considered the bottom. When a new subobject is added, it is always added to the top.

An empty RRO with no subobjects is considered illegal.

Two kinds of subobjects are currently defined.

4.4.1.1. Subobject 1: IPv4 address



Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

A 32-bit unicast, host address. Any network-reachable interface address is allowed here. Illegal addresses, such as certain loopback addresses, SHOULD NOT be used.

Prefix length

32

Flags

0x01 Local protection available

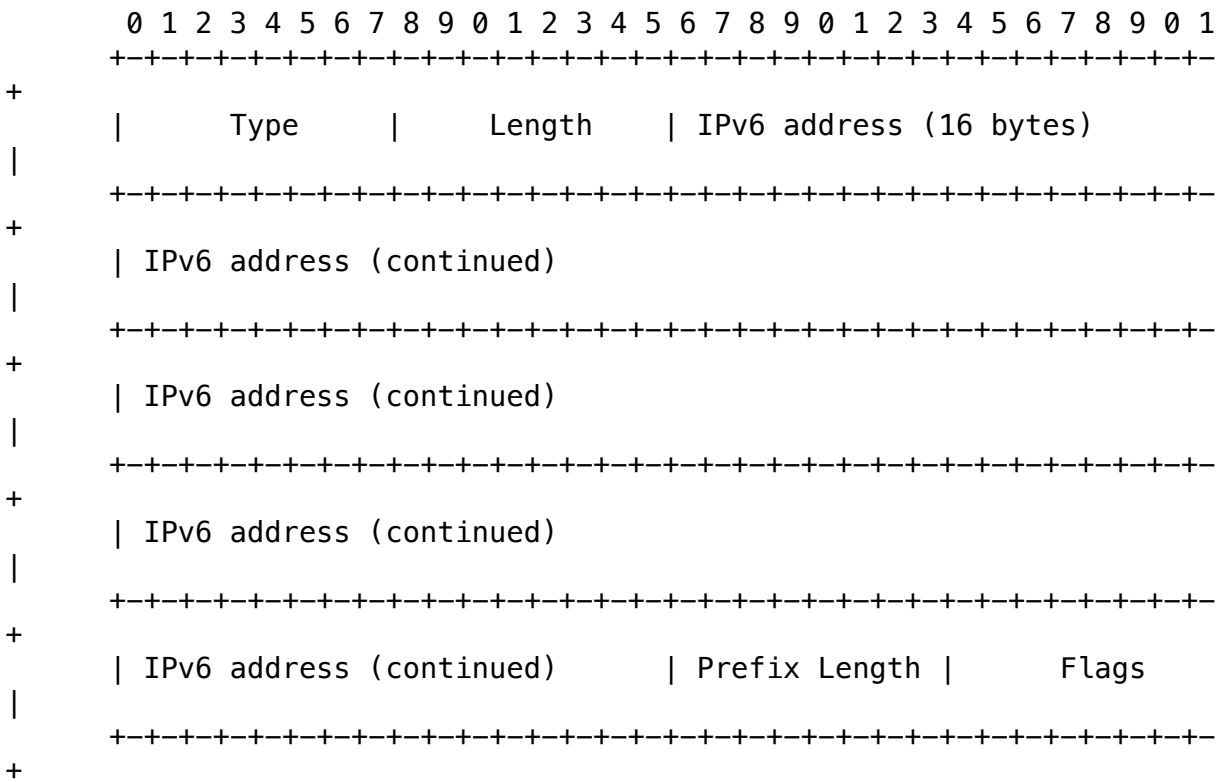
Indicates that the link downstream of this node is protected via a local repair mechanism. This flag can only be set if the Local protection flag was set in the SESSION_ATTRIBUITE object of the cooresponding Path nessage.

0x02 Local protection in use

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

4.4.1.2. Subobject 2: IPv6 address

0 1 2 3



Type

0x02 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

A 128-bit unicast host address.

Prefix length

128

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_ATTRIBUITE object of the cooresponding Path nessage.

0x02 Local protection in use

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

4.4.2. Applicability

Only the procedures for use in unicast sessions are defined here.

There are three possible uses of RRO in RSVP. First, an RRO can function as a loop detection mechanism to discover L3 routing loops, or loops inherent in the explicit route. The exact procedure for doing so is described later in this document.

Second, an RRO collects up-to-date detailed path information hop-by-hop about RSVP sessions, providing valuable information to the sender or receiver. Any path change (due to network topology changes) will be reported.

Third, RRO syntax is designed so that, with minor changes, the whole object can be used as input to the EXPLICIT_ROUTE object. This is useful if the sender receives RRO from the receiver in a Resv message, applies it to EXPLICIT_ROUTE object in the next Path message in order to "pin down session path".

4.4.3. Handling RRO

Typically, a node initiates an RSVP session by adding the RRO to the Path message. The initial RRO contains only one subobject - the sender's IP addresses.

When a Path message containing an RRO is received by an intermediate router, the router stores a copy of it in the Path State Block. The RRO is then used in the next Path refresh event for formatting Path messages. When a new Path message is to be sent, the router adds a new subobject to the RRO and appends the resulting RRO to the Path message before transmission.

The newly added subobject MUST be this router's IP address. The address to be added SHOULD be the interface address of the outgoing Path messages. If there are multiple addresses to choose from, the decision is a local matter. However, it is RECOMMENDED that the same address be chosen consistently.

If the newly added subobject causes the RRO to be too big to fit in a Path (or Resv) message, the RRO object SHALL be dropped from the message and message processing continues as normal. A PathErr (or ResvErr) message SHOULD be sent back to the sender (or receiver). An error code of "Notify" and an error value of "RRO too large for MTU" is used. The RRO object is included in the error message. If the receiver receives such a ResvErr, it SHOULD send a PathErr message with error code of "Notify" and an error value of "RRO notification". The RRO object is included in the error message.

A sender receiving either of these error values should remove the RRO from the Path message.

Nodes should resend the above PathErr or ResvErr message each n seconds where n is the greater of 15 and the refresh interval for the associated Path or RESV message. The node may apply limits and/or back-off timers to limit the number of messages sent.

An RSVP router can decide to send Path messages before its refresh time if the RRO in the next Path message is different from the previous one. This can happen if the contents of the RRO received from the previous hop router changes or if this RRO is newly added to (or deleted from) the Path message.

When the destination node of an RSVP session receives a Path message with an RRO, this indicates that the sender node needs route recording. The destination node initiates the RRO process by adding an RRO to Resv messages. The processing mirrors that of the Path messages. The only difference is that the RRO in a Resv message records the path information in the reverse direction.

Note that each node along the path will now have the complete route from source to destination. The Path RRO will have the route from the source to this node; the Resv RRO will have the route from this node to the destination. This is useful for network management.

A received Path message without an RRO indicates that the sender node no longer needs route recording. Subsequent Resv messages SHALL NOT contain an RRO.

4.4.4. Loop Detection

As part of processing an incoming RRO, an intermediate router looks into all subobjects contained within the RRO. If the router determines that it is already in the list, a forwarding loop

exists.

   An RSVP session is loop-free if downstream nodes receive Path
   messages or upstream nodes receive Resv messages with no routing
   loops detected in the contained RRO.

   There are two broad classifications of forwarding loops. The first
   class is the transient loop, which occurs as a normal part of
   operations as L3 routing tries to converge on a consistent
forwarding
   path for all destinations. The second class of forwarding loop is
   the permanent loop, which normally results from network mis-
   configuration.

   The action performed by a node on receipt of an RRO depends on the
   message type in which the RRO is received.

   For Path messages containing a forwarding loop, the router builds
and
   sends a "Routing problem" PathErr message, with the error value
"loop
   detected," and drops the Path message. Until the loop is
eliminated,
   this session is not suitable for forwarding data packets. How the
   loop eliminated is beyond the scope of this document.

   For Resv messages containing a forwarding loop, the router simply
   drops the message. Resv messages should not loop if Path messages
do
   not loop.

4.4.5. Non-support of RRO

   The RRO object is to be used only when all routers along the path
   support RSVP and the RRO object. The RRO object is assigned a
class
   value of the form 0bbbbbbb. RSVP routers that do not support the
   object will therefore respond with an "Unknown Object Class" error.

4.5. Error Codes for ERO and RRO

In the processing described above, certain errors must be reported as either a "Routing Problem" or "Notify". The value of the "Routing Problem" error code is 24; the value of the "Notify" error code is 25.

The following defines error values for the Routing Problem Error Code:

Value	Error:
1	Bad EXPLICIT_ROUTE object
2	Bad strict node
3	Bad loose node
4	Bad initial subobject
5	No route available toward destination
6	RRO syntax error detected
7	RRO indicated routing loops
8	MPLS being negotiated, but a non-RSVP-capable router stands in the path
9	MPLS label allocation failure
10	Unsupported L3PID

For the Notify Error Code, the 16 bits of the Error Value field are:

ss00 cccc cccc cccc

The high order bits are as defined under Error Code 1. (See [1]).

When ss = 00, the following subcode is defined:

```
          1    RRO too large for MTU
          2    RRO notification
```


4.6. Session, Sender Template, and Filter Spec Objects

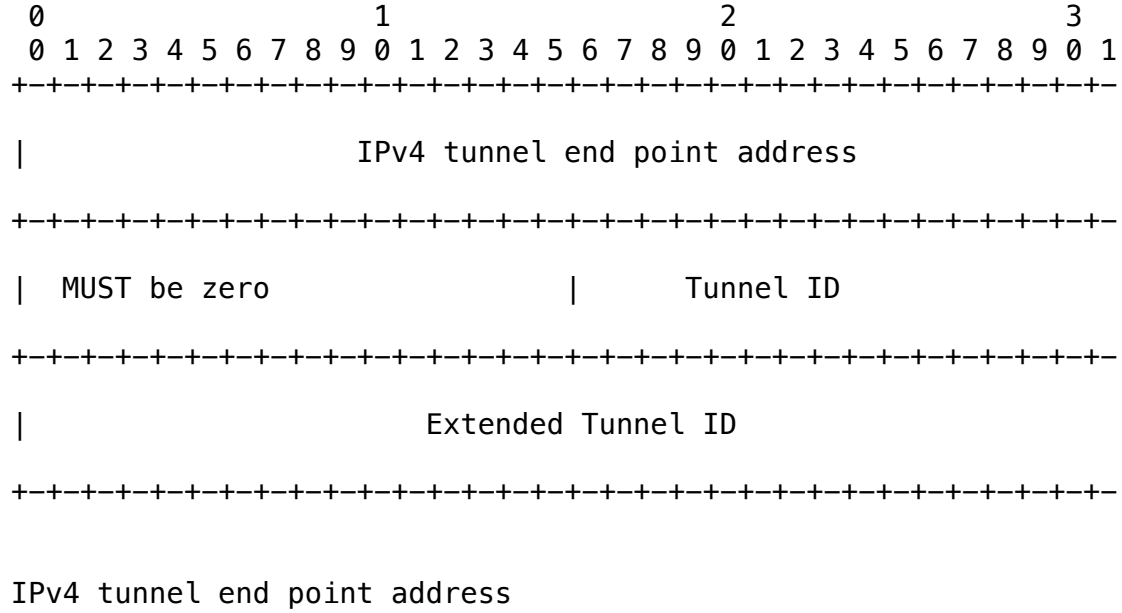
New C-Types are defined for the SESSION, SENDER_TEMPLATE and FILTER_SPEC objects.

The LSP_TUNNEL objects have the following format:

4.6.1. Session Object

4.6.1.1. LSP_TUNNEL_IPv4 Session Object

Class = SESSION, LSP_TUNNEL_IPv4 C-Type = 7



IPv4 tunnel end point address

IPv4 address of the egress node for the tunnel.

Tunnel ID

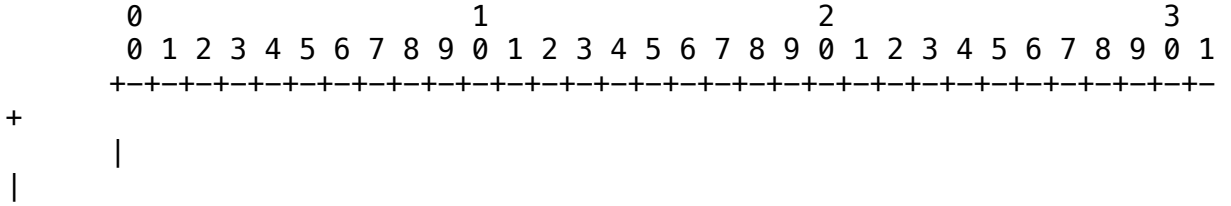
A 16-bit identifier used in the SESSION that remains constant over the life of the tunnel.

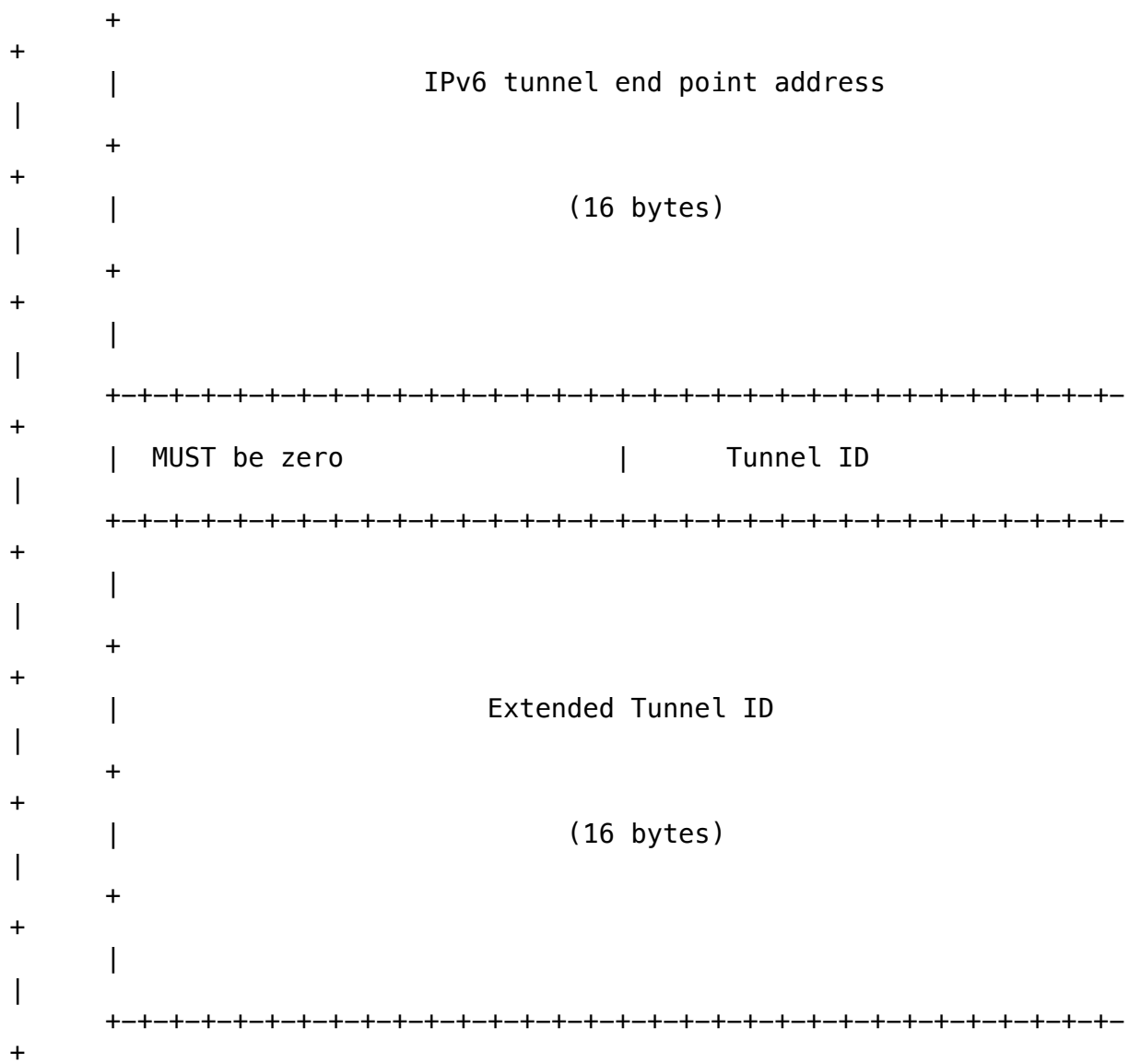
Extended Tunnel ID

A 32-bit identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv4 address here as a globally unique identifier.

4.6.1.2. LSP_TUNNEL_IPv6 Session Object

Class = SESSION, LSP_TUNNEL_IPv6 C_Type = 8





IPv6 tunnel end point address

IPv6 address of the egress node for the tunnel.

Tunnel ID

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

Extended Tunnel ID

A 16-byte identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all

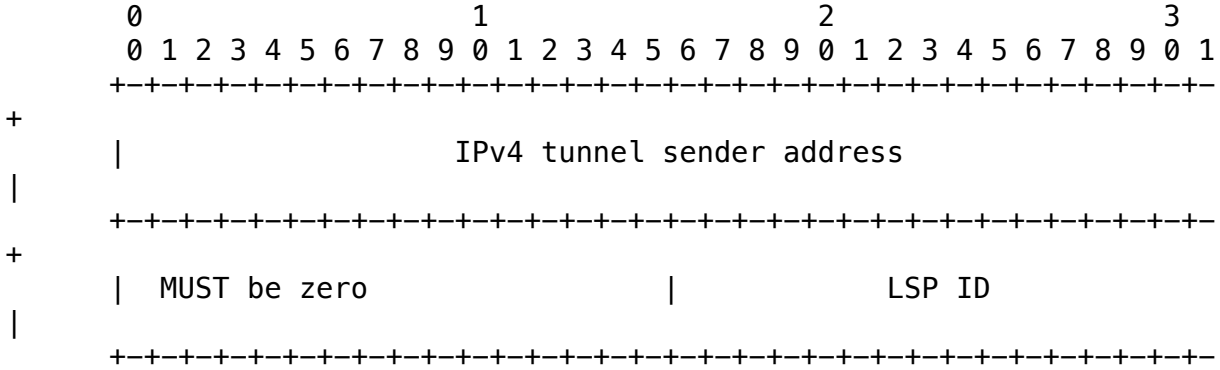
zeros.

Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv6 address here as a globally unique identifier.

4.6.2. Sender Template Object

4.6.2.1. LSP_TUNNEL_IPv4 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv4 C-Type = 7



IPv4 tunnel sender address

IPv4 address for a sender node

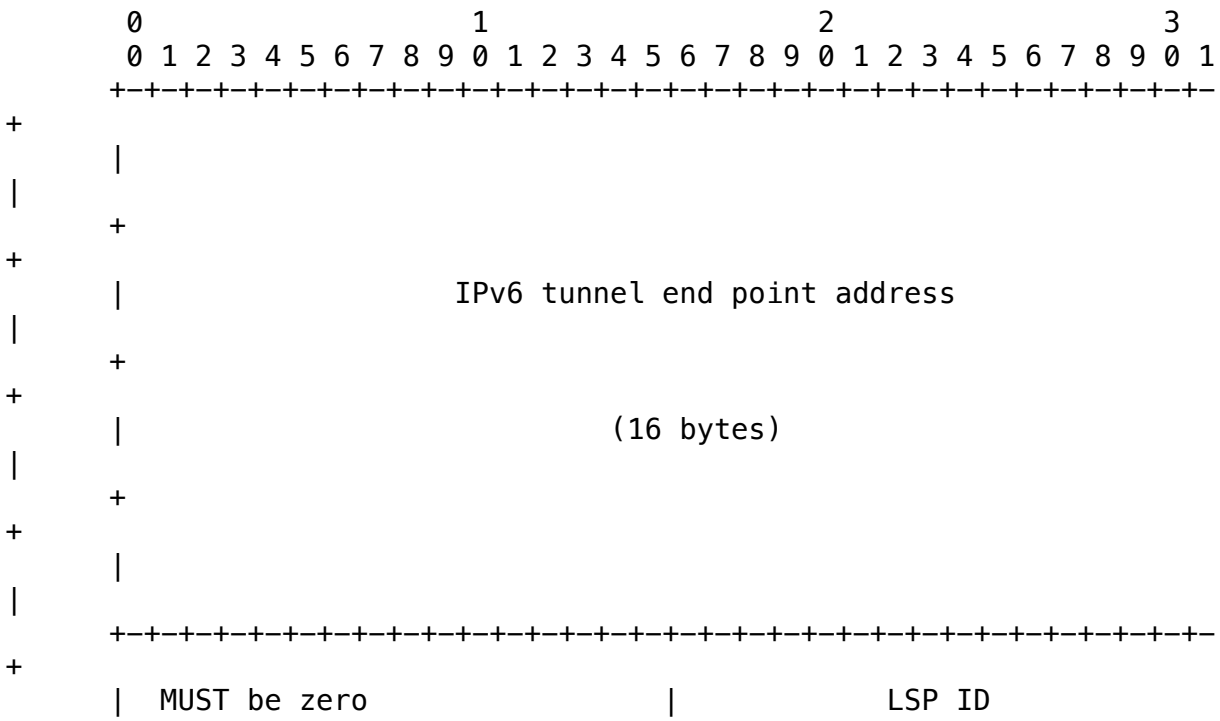
LSP ID

A 16-bit identifier used in the SENDER_TEMPLATE and the

FILTER_SPEC that can be changed to allow a sender to share
resources with itself.

4.6.2.2. LSP_TUNNEL_IPv6 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv6 C_Type = 8



IPv6 tunnel sender address

IPv6 address for a sender node

LSP ID

A 16-bit identifier used in the SENDER_TEMPLATE and the FILTER_SPEC that can be changed to allow a sender to share resources with itself.

4.6.3. Filter Specification Object

4.6.3.1. LSP_TUNNEL_IPv4 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv4 C-Type = 7

The format of the LSP_TUNNEL_IPv4 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv4 SENDER_TEMPLATE object.

4.6.3.2. LSP_TUNNEL_IPv6 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv6 C_Type = 8

The format of the LSP_TUNNEL_IPv6 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv6 SENDER_TEMPLATE object.

4.6.4. Reroute Procedure

This section describes how to setup a tunnel that is capable of maintaining resource reservations (without double counting) while it is being rerouted or while it is attempting to increase its bandwidth. In the initial Path message, the ingress node forms a SESSION object, assigns a Tunnel_ID, and places its IPv4 address in the Extended_Tunnel_ID It also forms a SENDER_TEMPLATE and assigns a LSP_ID. Tunnel setup then proceeds according to the normal procedure.

On receipt of the Path message, the egress node sends a Resv message with the STYLE Shared Explicit toward the ingress node.

When an ingress node with an established path wants to change that path, it forms a new Path message as follows. The existing SESSION object is used. In particular the Tunnel_ID and Extended_Tunnel_ID are unchanged. The ingress node picks a new LSP_ID to form a new

SENDER_TEMPLATE. It creates an EXPLICIT_ROUTE object for the new route. The new Path message is sent. The ingress node refreshes both the old and new path messages

The egress node responds with a Resv message with an SE flow

descriptor formatted as:

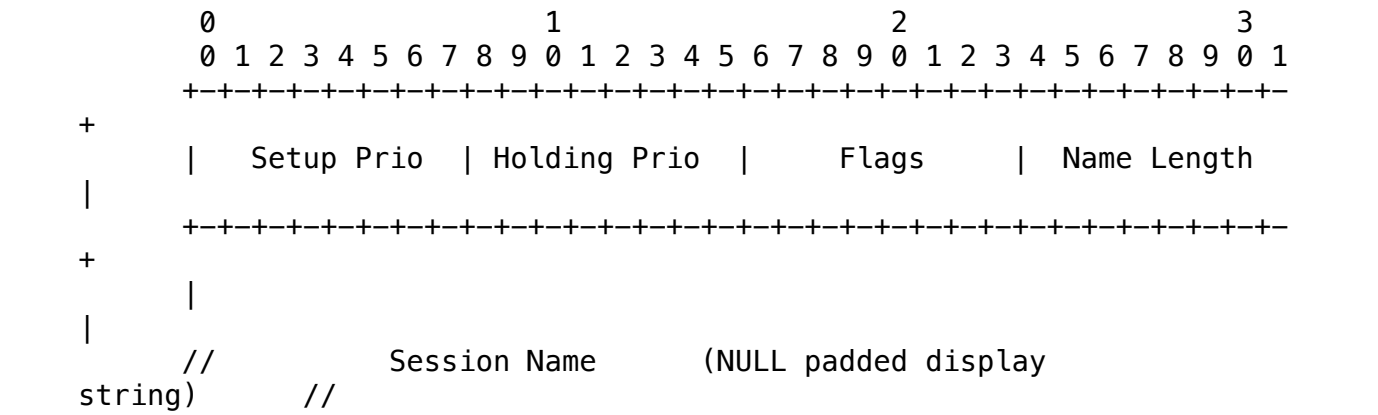
<FLOWSPEC><old_FILTER_SPEC><old_LABEL_OBJECT><new_FILTER_SPEC>
<new_LABEL_OBJECT>

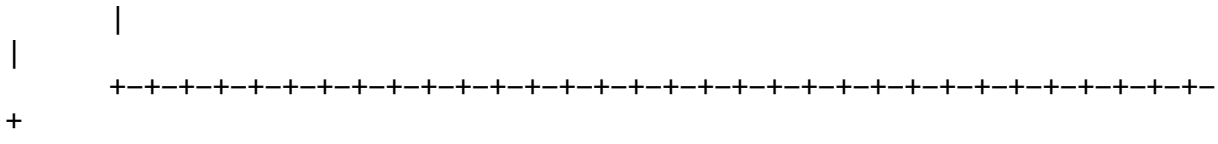
(Note that if the PHOPs are different, then two messages are sent each with the appropriate FILTER_SPEC and LABEL_OBJECT.)

When the ingress node receives the Resv Message(s), it may begin using the new route. It SHOULD send a PathTear message for the old route.

4.7. Session Attribute Object

The Session Attribute Class is 207. One C_Type is defined,
LSP_TUNNEL, C-Type = 7. The format of the LSP_TUNNEL Session
Attribute Object is as follows:





Flags

0x01 Local protection

This flag permits transit routers to use a local repair mechanism which may result in violation of the explicit route object. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

0x02 Merging permitted

This flag permits transit routers to merge this

session
with other RSVP sessions for the purpose of reducing
resource overhead on downstream transit routers,
thereby
providing better network scalability.

0x04 Ingress node may reroute

This flag indicates that the tunnel ingress node may
choose to reroute this tunnel without tearing it down.
A tunnel egress node SHOULD use the SE Style when
responding with a Resv message.

Setup Priority

The priority of the session with respect to taking
resources,
in the range of 0 to 7. The value 0 is the highest
priority.
The Setup Priority is used in deciding whether this session
can
preempt another session.

Holding Priority

The priority of the session with respect to holding
resources,
in the range of 0 to 7. The value 0 is the highest
priority.
Holding Priority is used in deciding whether this session
can
be preempted by another session.

Name Length

The length of the display string before padding, in bytes.

Session Name

A null padded string of characters.

The support of setup and holding priorities is OPTIONAL. A node
can

recognize this information but be unable to perform the requested operation. The node SHOULD pass the information downstream unchanged.

As noted above, preemption is implemented by two priorities. The Setup Priority is the priority for taking resources. The Holding Priority is the priority for holding a resource. Specifically, the Holding Priority is the priority at which resources assigned to this session will be reserved. The Setup Priority SHOULD never be higher than the Holding Priority for a given session.

The setup and holding priorities are directly analogous to the preemption and defending priorities as defined in [3]. While the interaction of these two objects is ultimately a matter of policy, the following default interaction is recommended.

When both objects are present, the preemption priority policy element is used. A mapping between the priority spaces is defined as follows. A session attribute priority S is mapped to a preemption priority P by the formula P = 2^(14-2S). The reverse mapping is shown in the following table.

Preemption Priority	Session Attribute Priority
0 - 3	7
4 - 15	6
16 - 63	5
64 - 255	4
256 - 1023	3
1024 - 4095	2
4096 - 16383	1
16384 - 65535	0

When a new reservation is considered for admission, the bandwidth requested is compared with the bandwidth available at the priority specified in the Setup Priority. The bandwidth available at a particular Setup Priority is the unused bandwidth plus the bandwidth reserved at all Holding Priorities lower than the Setup Priority.

If the requested bandwidth is not available a PathErr message is returned with an Error Code of 01, Admission Control Failure, and an

Error Value of 0x0002. The first 0 in the Error Value indicates a globally defined subcode and is not informational. The 002 indicates "requested bandwidth unavailable".

If the requested bandwidth is less than the unused bandwidth then processing is complete. If the requested bandwidth is available, but is in use by lower priority sessions, then lower priority sessions

(beginning with the lowest priority) can be pre-empted to free the necessary bandwidth.

When pre-emption is supported, each pre-empted reservation triggers a TC_Preempt() upcall to local clients, passing a subcode that indicates the reason. A ResvErr and/or PathErr with the code "Policy Control failure" SHOULD be sent toward the downstream receivers and upstream senders.

The support of local-protection is OPTIONAL. A node may recognize the local-protection Flag but may be unable to perform the requested operation. In this case, the node SHOULD pass the information downstream unchanged.

The support of merging is OPTIONAL. A node may recognize the Merge Flag but may be unable to perform the requested operation. In this case, the node SHOULD pass the information downstream unchanged.

If a Path message contains multiple SESSION_ATTRIBUTE objects, only the first SESSION_ATTRIBUTE object is meaningful. Subsequent SESSION_ATTRIBUTE objects can be ignored and need not be forwarded.

The contents of the Session Name field are a string, typically of displayable characters. The Length MUST always be a multiple of 4 and MUST be at least 8. For an object length that is not a multiple of 4, the object is padded with trailing NULL characters. The Name

Length field contains the actual string length.

All RSVP routers, whether they support the SESSION_ATTRIBUTE object or not, SHALL forward the object unmodified. The presence of non-RSVP routers anywhere between senders and receivers has no impact on this object.

4.8. Tspec and Flowspec Object for Class-of-Service Service

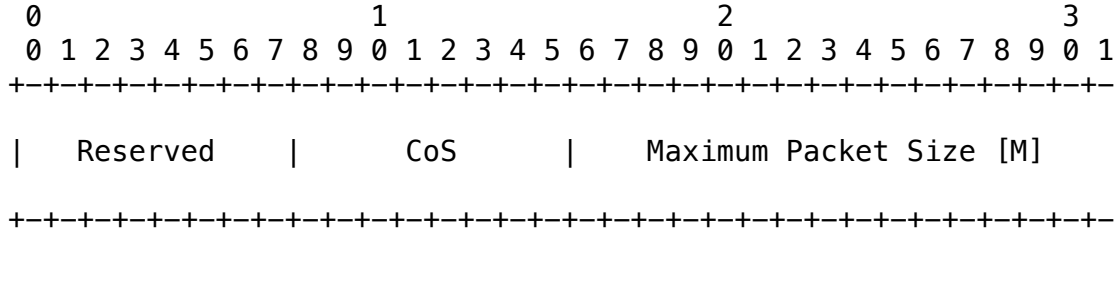
An LSP may not need bandwidth reservations or QoS guarantees. Such LSPs can be used to deliver best-effort traffic, even if RSVP is used for setting up LSPs. When resources do not have to be allocated to the LSP, the Class-of-Service service SHOULD be used.

The Class-of-Service FLOWSPEC allows indication of a Class of Service (CoS) value that should be used when handling data packets associated with the request.

The same format is used both for SENDER_TSPEC object and FLOWSPEC objects. The formats are:

Class-of-Service SENDER_TSPEC object: Class = 12, C-Type = 3

Class-of-Service FLOWSPEC object: Class = 9, C-Type = 3



Reserved

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

CoS

Indicates the Class of Service (CoS) of the data traffic associated with the request. A value of zero (0) indicates that associated traffic is "Best-Effort". Specifically no service assurances are being requested from the network. The intent is to enable networks to support the IP ToS Octet as defined in RFC1349 [7]. It is noted that there is ongoing work within the IETF to update the use of the IP ToS Octet. In particular, RFC2474 [8], obsoletes RFC1349. However at this time the new uses of this field (now termed the DS byte) are still being defined. Non-zero values have local significance. The translation from a specific value to an allocation is a local administrative decision.

M

This parameter is set in Resv messages by the receiver based on information in arriving RSVP SENDER_TSPEC objects. For shared reservations, the smallest value received across all associated senders is used. When the object is contained in Path messages, this parameter is updated at each hop with the lesser of the received value and the MTU of the outgoing interface.

There is no Adspec associated with the Class-of-Service SENDER_TSPEC.
Either the Adspec is omitted or an int-serv adspec with only the Default General Characterization Parameters is used.

5. Hello Extension

The RSVP Hello extension enables RSVP nodes to detect when a neighboring node is not reachable. The mechanism provides node to node failure detection. When such a failure is detected it is handled much the same as a link layer communication failure. This mechanism is intended to be used when notification of link layer failures is not available and unnumber links are not used, or when the failure detection mechanisms provided by the link layer are not sufficient for timely node failure detection.

It should be noted that node failure detection is not the same as a link failure detection mechanism, particularly in the case of multiple parallel unnumbered links.

The Hello extension is specifically designed so that one side can use the mechanism while the other side does not. Neighbor failure detection may be initiated at any time. This includes when neighbors first learn about each other, or just when neighbors are sharing Resv or Path state.

The Hello extension is composed of a Hello message, a HELLO REQUEST object and a HELLO ACK object. Hello processing between two neighbors supports independent selection of, typically configured, failure detection intervals. Each neighbor can autonomously issue HELLO REQUEST objects. Each request is answered by an acknowledgment. Hello Messages also contain enough information so that one neighbor can suppress issuing hello requests and still perform neighbor failure detection. A Hello message may be included

as a sub-message within a bundle message.

Neighbor failure detection is accomplished by collecting and storing a neighbor's "instance" value. If a change in value is seen or if the neighbor is not properly reporting the locally advertised value, then the neighbor is presumed to have reset. When a neighbor's value is seen to change or when communication is lost with a neighbor, then the instance value advertised to that neighbor is also changed. The HELLO objects provide a mechanism for polling for and providing an instance value. A poll request also includes the sender's instance value. This allows the receiver of a poll to optionally treat the poll as an implicit poll response. This optional handling is an optimization that can reduce the total number of polls and responses processed by a pair of neighbors. In all cases, when both sides support the optimization the result will be only one set of polls and responses per failure detection interval. Depending on selected intervals, the same benefit can occur even when only one neighbor supports the optimization.

5.1. Hello Message Format

Hello Messages are always sent between two RSVP neighbors. The IP source address is the IP address of the sending node. The IP destination address is the IP address of the neighbor node.

The HELLO mechanism is intended for use between immediate neighbors. When HELLO messages are being the exchanged between immediate neighbors, the IP TTL field of all outgoing HELLO messages SHOULD be set to 1.

The Hello message format is as follows:

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

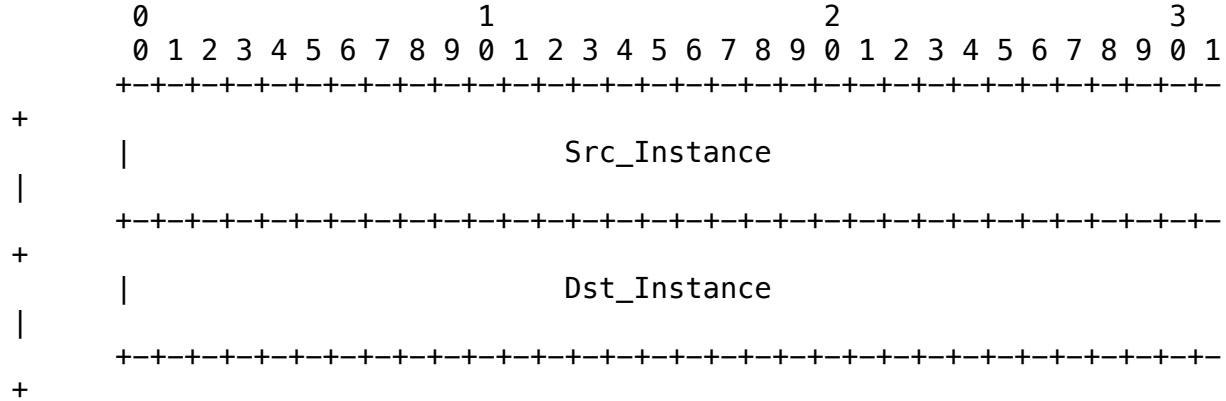
For Hello messages, the Msg Type field of the Common Header MUST be set to 14.

5.2. HELLO Object

HELLO Class = 22

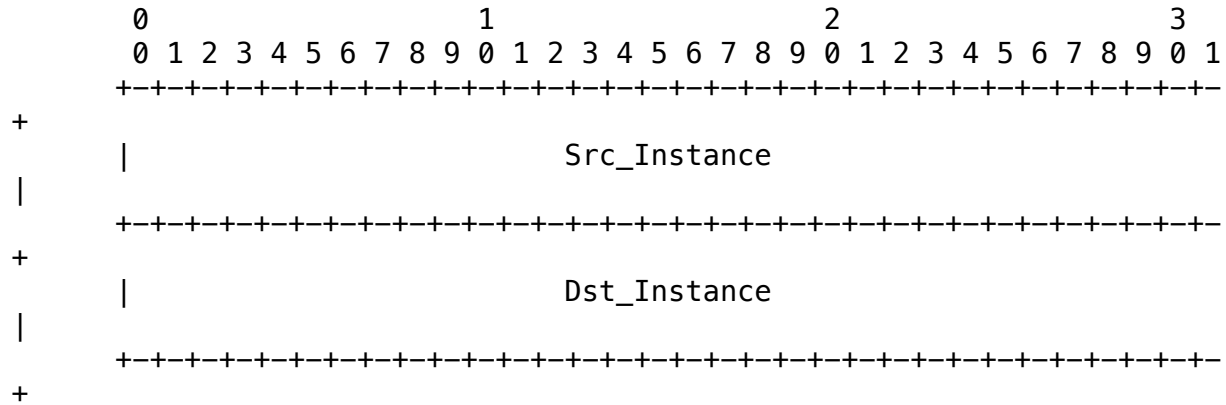
HELLO REQUEST object

Class = HELLO Class, C_Type = 1



HELLO ACK object

Class = HELLO Class, C_Type = 2



Src_Instance: 32 bits

a 32 bit value that represents the sender's instance. The advertiser maintains a per neighbor representation/value. This value MUST change when the sender is reset, when the node reboots, or when communication is lost to the neighboring node and otherwise remains the same. This field MUST NOT be set to zero (0).

Dst_Instance: 32 bits

The most recently received Src_Instance value received from the neighbor. This field MUST be set to zero (0) when no value has ever been seen from the neighbor.

5.3. Hello Message Usage

The Hello Message is completely optional. All messages may be ignored by nodes which do not wish to participate in Hello message processing. The balance of this section is written assuming that the receiver as well as the sender is participating. In particular, the use of MUST and SHOULD with respect to the receiver applies only to a

A node periodically generates a Hello message containing a HELLO REQUEST object for each neighbor who's status is being tracked. The periodicity is governed by the hello_interval. This value MAY be configured on a per neighbor basis. The default value is 5 ms.

When generating a message containing a HELLO REQUEST object, the sender fills in the Src_Instance field with a value representing it's per neighbor instance. This value MUST NOT change while the agent is exchanging Hellos with the corresponding neighbor. The sender also

fills in the Dst_Instance field with the Src_Instance value most recently received from the neighbor. If no value has ever been received from the neighbor, a value of zero (0) is used. The generation of a message SHOULD be suppressed when a HELLO REQUEST object was received from the destination node within the prior hello_interval interval.

On receipt of a message containing a HELLO REQUEST object, the receiver MUST generate a Hello message containing a HELLO ACK object.
The receiver SHOULD also verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the value differs, then a node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO REQUEST object SHOULD also verify that the neighbor is reflecting back the receiver's Instance value. This is done by comparing the received Dst_Instance field with the

Src_Instance field value most recently transmitted to that neighbor.
If the neighbor continues to advertise a wrong non-zero value after a configured number of intervals, then the node MUST treat the neighbor as if communication has been lost.

On receipt of a message containing a HELLO ACK object, the receiver MUST verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the value differs, then the node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO ACK object MUST also verify that the neighbor is reflecting back the receiver's Instance value. If the neighbor advertises a wrong value in the Dst_Instance field, then a node MUST treat the neighbor as if communication has been lost.

If no Instance values are received, via either REQUEST or ACK objects, from a neighbor within a configured number of hello_intervals, then a node MUST presume that it cannot communicate with the neighbor. The default for this number is 3.5.

When communication is lost or presumed to be lost as described above, a node MAY re-initiate HELLOs. If a node does re-initiate it MUST use a Src_Instance value different than the one advertised in the previous HELLO message. This new value MUST continue to be advertised to the corresponding neighbor until a reset or reboot occurs, or until another communication failure is detected. If a new instance value has not been received from the neighbor, then the node MUST advertise zero in the Dst_instance value field.

5.4. Multi-Link Considerations

As previously noted, the Hello extension is targeted at detecting node failures not per link failures. When there is only one link between neighboring nodes or when all links between a pair of nodes fail, the distinction between node and link failures is not really meaningful and handling of such failures has already been covered. When there are multiple links shared between neighbors, there are special considerations. When the links between neighbors are numbered, then Hellos MUST be run on each link and the previously described mechanisms apply.

When the links are unnumbered, link failure detection MUST be provided by some means other than Hellos. Each node SHOULD use a single Hello exchange with the neighbor. The case where all links have failed, is the same as the no received value case mentioned in the previous section.

5.5. Compatibility

The Hello extension does not affect the processing of any other RSVP message. The only effect is to allow a link (node) down event to be declared sooner than it would have been. RSVP response to that condition is unchanged.

The Hello extension is fully backwards compatible. The Hello class is assigned a class value of the form 0bbbbbbb. Depending on the implementation, implementations that do not support the extension will either silently discard Hello messages or will respond with an "Unknown Object Class" error. In either case the sender will fail to see an acknowledgment for the issued Hello.

6. Security Considerations

In principle these extentions to RSVP pose no security exposures over and above RFC 2205[1]. However, there is a slight change in the trust model. Traffic sent on a normal RSVP session can be filtered according to source and destination addresses as well as port numbers. In this specification, filtering occurs only on the basis of an incoming label. For this reason an administration may which to limit the domain over which LSP tunnels can be established. This can be accomplished by setting filters on various ports to deny action on a RSVP path message with a SESSION object of type LSP_TUNNEL_IPv4 (7) or LSP_TUNNEL_IPv4 (8).

7. IANA Considerations

The responsible Internet authority (presently called the IANA) assigns values to RSVP protocol parameters. With the current document an EXPLICIT_ROUTE object and a ROUTE_RECORD object are defined. Each of these objects contain subobjects. This section defines the rules for the assignment of subobject numbers. This section uses the terminology of BCP 26 "Guidelines for Writing an IANA Considerations Section in RFCs".

EXPLICIT_ROUTE Subobject Type

EXPLICIT_ROUTE Subobject Type is a 7-bit number that identifies

the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

ROUTE_RECORD Subobject Type

ROUTE_RECORD Subobject Type is an 8-bit number that identifies the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

8. Intellectual Property Considerations

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

9. Acknowledgments

This document contains ideas as well as text that have appeared in previous Internet Drafts. The authors of the current draft wish to thank the authors of those drafts. They are Steven Blake, Bruce Davie, Roch Guerin, Sanjay Kamat, Yakov Rekhter, Eric Rosen, and Arun Viswanathan. We also wish to thank Bora Akyol, Yoram Bernet and Alex Mondrus for their comments on this draft.

10. References

[1] Braden, R. et al., "Resource ReSerVation Protocol (RSVP) -- Version 1, Functional Specification", RFC 2205, September 1997.

[2] Rosen, E. et al., "Multiprotocol Label Switching Architeture",

Internet Draft, draft-ietf-mpls-arch-06.txt, August 1999.

[3] Awduche, D. et al., "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[4] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.

[5] Rosen, E. et al., "MPLS Label Stack Encoding", Internet Draft, draft-ietf-mpls-label-encaps-07.txt, September 1999.

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

[7] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[8] Nichols, K. et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[9] Herzog, S., "Signaled Preemption Priority Policy Element", draft-ietf-rap-signaled-priority-04.txt, September 1999.

[10] Awduche, D. et al., "Applicability Statement for Extensions to RSVP for LSP-Tunnels", draft-ietf-mpls-rsvp-tunnel-applicability-00.txt, September 1999.

11. Authors' Addresses

Daniel O. Awduche
UUNET (MCI Worldcom), Inc
3060 Williams Drive
Fairfax, VA 22031
Voice: +1 703 208 5277
Email: awduche@uu.net

Lou Berger
LabN Consulting, LLC
Voice: +1 301 468 9228
Email: lberger@labn.net

Der-Hwa Gan
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
Voice: +1 650 526
Email: dhg@juniper.net

Tony Li
Procket Networks
3910 Freedom Circle, Ste. 102A
Santa Clara CA 95054
Email: tony1@home.net

Vijay Srinivasan
Torrent Networking Technologies Corp.
3000 Aerial Center Parkway, Suite 140
Morrisville, NC 27560
Voice: +1 919 468 8466 ext. 236
Email: vijay@torrentnet.com

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

Network Working Group Daniel O. Awduche
Internet Draft UUNET (MCI Worldcom), Inc.
Expiration Date: February 2001
Lou Berger
LabN Consulting, LLC
Der-Hwa Gan
Juniper Networks, Inc.
Tony Li
Procket Networks, Inc.
Vijay Srinivasan
Cosine Communications, Inc.
George Swallow
Cisco Systems, Inc.

August 2000

RSVP-TE: Extensions to RSVP for LSP Tunnels

draft-ietf-mpls-rsvp-lsp-tunnel-07.txt

Status of this Memo

and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

To view the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in an Internet-Drafts Shadow Directory, see http://www.ietf.org/shadow.html.

Abstract

This document describes the use of RSVP, including all the necessary extensions, to establish label-switched paths (LSPs) in MPLS. Since the flow along an LSP is completely identified by the label applied

at the ingress node of the path, these paths may be treated as tunnels. A key application of LSP tunnels is traffic engineering with MPLS as specified in [3].

We propose several additional objects that extend RSVP, allowing the establishment of explicitly routed label switched paths using RSVP as a signaling protocol. The result is the instantiation of label-switched tunnels which can be automatically routed away from network failures, congestion, and bottlenecks.

Contents

1. Introduction

Section 2.9 of the MPLS architecture [2] defines a label distribution protocol as a set of procedures by which one Label Switched Router (LSR) informs another of the meaning of labels used to forward traffic between and through them. The MPLS architecture does not assume a single label distribution protocol. This document is a specification of extensions to RSVP for establishing label switched paths (LSPs) in Multi-protocol Label Switching (MPLS) networks.

Several of the new features described in this document were motivated by the requirements for traffic engineering over MPLS (see [3]). In particular, the extended RSVP protocol supports the instantiation of explicitly routed LSPs, with or without resource reservations. It

also supports smooth rerouting of LSPs, preemption, and loop detection.

The LSPs created with RSVP can be used to carry the "Traffic Trunks" described in [3]. The LSP which carries a traffic trunk and a traffic trunk are distinct though closely related concepts. For example, two LSPs between the same source and destination could be load shared to carry a single traffic trunk. Conversely several traffic trunks could be carried in the same LSP if, for instance, the LSP were capable of carrying several service classes. The applicability of these extensions is discussed further in [10].

Since the traffic that flows along a label-switched path is defined by the label applied at the ingress node of the LSP, these paths can be treated as tunnels, tunneling below normal IP routing and filtering mechanisms. When an LSP is used in this way we refer to it as an LSP tunnel.

LSP tunnels allow the implementation of a variety of policies related to network performance optimization. For example, LSP tunnels can be automatically or manually routed away from network failures, congestion, and bottlenecks. Furthermore, multiple parallel LSP tunnels can be established between two nodes, and traffic between the two nodes can be mapped onto the LSP tunnels according to local policy. Although traffic engineering (that is, performance optimization of operational networks) is expected to be an important application of this specification, the extended RSVP protocol can be used in a much wider context.

The purpose of this document is to describe the use of RSVP to establish LSP tunnels. The intent is to fully describe all the objects, packet formats, and procedures required to realize interoperable implementations. A few new objects are also defined that enhance management and diagnostics of LSP tunnels.

The document also describes a means of rapid node failure detection via a new HELLO message.

All objects and messages described in this specification are optional with respect to RSVP. This document discusses what happens when an object described here is not supported by a node.

Throughout this document, the discussion will be restricted to unicast label switched paths. Multicast LSPs are left for further study.

1.1. Background

Hosts and routers that support both RSVP [1] and Multi-Protocol Label Switching [2] can associate labels with RSVP flows. When MPLS and RSVP are combined, the definition of a flow can be made more flexible. Once a label switched path (LSP) is established, the traffic through the path is defined by the label applied at the ingress node of the LSP. The mapping of label to traffic can be accomplished using a number of different criteria. The set of packets that are assigned the same label value by a specific node are said to belong to the same forwarding equivalence class (FEC) (see [2]), and effectively define the "RSVP flow." When traffic is mapped onto a label-switched path in this way, we call the LSP an "LSP Tunnel". When labels are associated with traffic flows, it becomes possible for a router to identify the appropriate reservation state for a packet based on the packet's label value.

The signaling protocol model uses downstream-on-demand label distribution. A request to bind labels to a specific LSP tunnel is initiated by an ingress node through the RSVP Path message. For this purpose, the RSVP Path message is augmented with a LABEL_REQUEST object. Labels are allocated downstream and distributed (propagated upstream) by means of the RSVP Resv message. For this purpose, the RSVP Resv message is extended with a special LABEL object. The procedures for label allocation, distribution, binding, and stacking are described in subsequent sections of this document.

The signaling protocol model also supports explicit routing

capability. This is accomplished by incorporating a simple EXPLICIT_ROUTE object into RSVP Path messages. The EXPLICIT_ROUTE object encapsulates a concatenation of hops which constitutes the explicitly routed path. Using this object, the paths taken by label-switched RSVP-MPLS flows can be pre-determined, independent of conventional IP routing. The explicitly routed path can be administratively specified, or automatically computed by a suitable entity based on QoS and policy requirements, taking into

consideration the prevailing network state. In general, path computation can be control-driven or data-driven. The mechanisms, processes, and algorithms used to compute explicitly routed paths are beyond the scope of this specification.

One useful application of explicit routing is traffic engineering. Using explicitly routed LSPs, a node at the ingress edge of an MPLS domain can control the path through which traffic traverses from itself, through the MPLS network, to an egress node. Explicit routing can be used to optimize the utilization of network resources and enhance traffic oriented performance characteristics.

The concept of explicitly routed label switched paths can be generalized through the notion of abstract nodes. An abstract node is a group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node. Using this concept of abstraction, an explicitly routed LSP can be specified as a sequence of IP prefixes or a sequence of Autonomous Systems.

The signaling protocol model supports the specification of an explicit path as a sequence of strict and loose routes. The combination of abstract nodes, and strict and loose routes significantly enhances the flexibility of path definitions.

An advantage of using RSVP to establish LSP tunnels is that it

enables the allocation of resources along the path. For example, bandwidth can be allocated to an LSP tunnel using standard RSVP reservations and Integrated Services service classes [4].

While resource reservations are useful, they are not mandatory. Indeed, an LSP can be instantiated without any resource reservations whatsoever. Such LSPs without resource reservations can be used, for example, to carry best effort traffic. They can also be used in many other contexts, including implementation of fall-back and recovery policies under fault conditions, and so forth.

1.2. 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 [6].

The reader is assumed to be familiar with the terminology in [1], [2] and [3].

Abstract Node

A group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node.

Explicitly Routed LSP

An LSP whose path is established by a means other than normal IP routing.

Label Switched Path

The path created by the concatenation of one or more label switched hops, allowing a packet to be forwarded by swapping labels from an MPLS node to another MPLS node. For a more precise definition see [2].

LSP

A Label Switched Path

LSP Tunnel

An LSP which is used to tunnel below normal IP routing and/or filtering mechanisms.

Traffic Engineered Tunnel (TE Tunnel)

A set of one or more LSP Tunnels which carries a traffic trunk.

Traffic Trunk

A set of flows aggregated by their service class and then placed on an LSP or set of LSPs called a traffic engineered tunnel. For further discussion see [3].

2. Overview

2.1. LSP Tunnels and Traffic Engineered Tunnels

According to [1], "RSVP defines a 'session' to be a data flow with a particular destination and transport-layer protocol." However, when RSVP and MPLS are combined, a flow or session can be defined with greater flexibility and generality. The ingress node of an LSP can use a variety of means to determine which packets are assigned a particular label. Once a label is assigned to a set of packets, the label effectively defines the "flow" through the LSP. We refer to such an LSP as an "LSP tunnel" because the traffic through it is opaque to intermediate nodes along the label switched path.

New RSVP SESSION, SENDER_TEMPLATE, and FILTER_SPEC objects, called LSP_TUNNEL_IPv4 and LSP_TUNNEL_IPv6 have been defined to support the LSP tunnel feature. The semantics of these objects, from the perspective of a node along the label switched path, is that traffic belonging to the LSP tunnel is identified solely on the basis of packets arriving from the PHOP or "previous hop" (see [1]) with the particular label value(s) assigned by this node to upstream senders to the session. In fact, the IPv4(v6) that appears in the object name only denotes that the destination address is an IPv4(v6) address. When we refer to these objects generically, we use the qualifier LSP_TUNNEL.

In some applications it is useful to associate sets of LSP tunnels. This can be useful during reroute operations or to spread a traffic trunk over multiple paths. In the traffic engineering application such sets are called traffic engineered tunnels (TE tunnels). To enable the identification and association of such LSP tunnels, two identifiers are carried. A tunnel ID is part of the SESSION object. The SESSION object uniquely defines a traffic engineered tunnel. The SENDER_TEMPLATE and FILTER_SPEC objects carry an LSP ID. The SENDER_TEMPLATE (or FILTER_SPEC) object together with the SESSION object uniquely identifies an LSP tunnel

2.2. Operation of LSP Tunnels

This section summarizes some of the features supported by RSVP as extended by this document related to the operation of LSP tunnels. These include: (1) the capability to establish LSP tunnels with or without QoS requirements, (2) the capability to dynamically reroute an established LSP tunnel, (3) the capability to observe the actual route traversed by an established LSP tunnel, (4) the capability to identify and diagnose LSP tunnels, (5) the capability to preempt an

established LSP tunnel under administrative policy control, and (6) the capability to perform downstream-on-demand label allocation, distribution, and binding. In the following paragraphs, these features are briefly described. More detailed descriptions can be found in subsequent sections of this document.

To create an LSP tunnel, the first MPLS node on the path -- that is, the sender node with respect to the path -- creates an RSVP Path message with a session type of LSP_TUNNEL_IPv4 or LSP_TUNNEL_IPv6 and inserts a LABEL_REQUEST object into the Path message. The LABEL_REQUEST object indicates that a label binding for this path is requested and also provides an indication of the network layer protocol that is to be carried over this path. The reason for this is that the network layer protocol sent down an LSP cannot be assumed to be IP and cannot be deduced from the L2 header, which simply identifies the higher layer protocol as MPLS.

If the sender node has knowledge of a route that has high likelihood of meeting the tunnel's QoS requirements, or that makes efficient use of network resources, or that satisfies some policy criteria, the node can decide to use the route for some or all of its sessions. To do this, the sender node adds an EXPLICIT_ROUTE object to the RSVP

Path message. The EXPLICIT_ROUTE object specifies the route as a sequence of abstract nodes.

If, after a session has been successfully established, the sender node discovers a better route, the sender can dynamically reroute the session by simply changing the EXPLICIT_ROUTE object. If problems are encountered with an EXPLICIT_ROUTE object, either because it causes a routing loop or because some intermediate routers do not support it, the sender node is notified.

By adding a RECORD_ROUTE object to the Path message, the sender node can receive information about the actual route that the LSP tunnel traverses. The sender node can also use this object to request notification from the network concerning changes to the routing path.
The RECORD_ROUTE object is analogous to a path vector, and hence can be used for loop detection.

Finally, a SESSION_ATTRIBUTE object can be added to Path messages to aid in session identification and diagnostics. Additional control information, such as setup and hold priorities, resource affinities (see [3]), and local-protection, are also included in this object.

Routers along the path may use the setup and hold priorities along with SENDER_TSPEC and any POLICY_DATA objects contained in Path messages as input to policy control. For instance, in the traffic engineering application, it is very useful to use the Path message as

a means of verifying that bandwidth exists at a particular priority along an entire path before pre-empting any lower priority reservations. If a Path message is allowed to progress when there are insufficient resources, the there is a danger that lower priority reservations downstream of this point will unnecessarily be pre-empted in a futile attempt to service this request.

When the EXPLICIT_ROUTE object (ERO) is present, the Path message is forwarded towards its destination along a path specified by the ERO.

Each node along the path records the ERO in its path state block. Nodes may also modify the ERO before forwarding the Path message. In this case the modified ERO SHOULD be stored in the path state block in addition to the received ERO.

The LABEL_REQUEST object requests intermediate routers and receiver nodes to provide a label binding for the session. If a node is incapable of providing a label binding, it sends a PathErr message with an "unknown object class" error. If the LABEL_REQUEST object is not supported end to end, the sender node will be notified by the first node which does not provide this support.

The destination node of a label-switched path responds to a LABEL_REQUEST by including a LABEL object in its response RSVP Resv message. The LABEL object is inserted in the filter spec list immediately following the filter spec to which it pertains.

The Resv message is sent back upstream towards the sender, following the path state created by the Path message, in reverse order. Note that if the path state was created by use of an ERO, then the Resv message will follow the reverse path of the ERO.

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, 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. The label sent upstream in the LABEL object is the label which this node will use to identify incoming traffic associated with this LSP tunnel. This label also serves as shorthand for the Filter Spec. The node can now update its "Incoming Label Map" (ILM), which is used to map incoming labeled packets to a "Next Hop Label Forwarding Entry" (NHLFE), see [2].

When the Resv message propagates upstream to the sender node, a label-switched path is effectively established.

2.3. Service Classes

This document does not restrict the type of Integrated Service requests for reservations. However, an implementation SHOULD support the Controlled-Load service [4] and the Class-of-Service service, see Section 4.8.

2.4. Reservation Styles

The receiver node can select from among a set of possible reservation styles for each session, and each RSVP session must have a particular style. Senders have no influence on the choice of reservation style. The receiver can choose different reservation styles for different LSPs.

An RSVP session can result in one or more LSPs, depending on the reservation style chosen.

Some reservation styles, such as FF, dedicate a particular reservation to an individual sender node. Other reservation styles, such as WF and SE, can share a reservation among several sender nodes. The following sections discuss the different reservation styles and their advantages and disadvantages. A more detailed discussion of reservation styles can be found in [1].

2.4.1. Fixed Filter (FF) Style

The Fixed Filter (FF) reservation style creates a distinct

reservation for traffic from each sender that is not shared by other senders. This style is common for applications in which traffic from each sender is likely to be concurrent and independent. The total amount of reserved bandwidth on a link for sessions using FF is the sum of the reservations for the individual senders.

Because each sender has its own reservation, a unique label is assigned to each sender. This can result in a point-to-point LSP between every sender/receiver pair.

2.4.2. Wildcard Filter (WF) Style

With the Wildcard Filter (WF) reservation style, a single shared reservation is used for all senders to a session. The total reservation on a link remains the same regardless of the number of senders.

A single multipoint-to-point label-switched-path is created for all senders to the session. On links that senders to the session share, a single label value is allocated to the session. If there is only one sender, the LSP looks like a normal point-to-point connection. When multiple senders are present, a multipoint-to-point LSP (a reversed tree) is created.

This style is useful for applications in which not all senders send traffic at the same time. A phone conference, for example, is an application where not all speakers talk at the same time. If, however, all senders send simultaneously, then there is no means of getting the proper reservations made. Either the reserved bandwidth on links close to the destination will be less than what is required or then the reserved bandwidth on links close to some senders will

be
greater than what is required. This restricts the applicability of WF for traffic engineering purposes.

Furthermore, because of the merging rules of WF, EXPLICIT_ROUTE objects cannot be used with WF reservations. As a result of this issue and the lack of applicability to traffic engineering, use of WF
is not considered in this document.

2.4.3. Shared Explicit (SE) Style

The Shared Explicit (SE) style allows a receiver to explicitly specify the senders to be included in a reservation. There is a single reservation on a link for all the senders listed. Because each sender is explicitly listed in the Resv message, different labels may be assigned to different senders, thereby creating separate LSPs.

SE style reservations can be provided using multipoint-to-point label-switched-path or LSP per sender. Multipoint-to-point LSPs may
be used when path messages do not carry the EXPLICIT_ROUTE object, or
when Path messages have identical EXPLICIT_ROUTE objects. In either
of these cases a common label may be assigned.

Path messages from different senders can each carry their own ERO, and the paths taken by the senders can converge and diverge at any point in the network topology. When Path messages have differing EXPLICIT_ROUTE objects, separate LSPs for each EXPLICIT_ROUTE object
must be established.

2.5. Rerouting Traffic Engineered Tunnels

One of the requirements for Traffic Engineering is the capability to reroute an established TE tunnel under a number of conditions, based on administrative policy. For example, in some contexts, an administrative policy may dictate that a given TE tunnel is to be rerouted when a more "optimal" route becomes available. Another important context when TE tunnel reroute is usually required is upon failure of a resource along the TE tunnel's established path. Under some policies, it may also be necessary to return the TE tunnel to its original path when the failed resource becomes re-activated.

In general, it is highly desirable not to disrupt traffic, or adversely impact network operations while TE tunnel rerouting is in progress. This adaptive and smooth rerouting requirement necessitates establishing a new LSP tunnel and transferring traffic from the old LSP tunnel onto it before tearing down the old LSP tunnel. This concept is called "make-before-break." A problem can arise because the old and new LSP tunnels might compete with each other for resources on network segments which they have in common. Depending on availability of resources, this competition can cause Admission Control to prevent the new LSP tunnel from being established. An advantage of using RSVP to establish LSP tunnels is that it solves this problem very elegantly.

To support make-before-break in a smooth fashion, it is necessary that on links that are common to the old and new LSPs, resources used by the old LSP tunnel should not be released before traffic is transitioned to the new LSP tunnel, and reservations should not be counted twice because this might cause Admission Control to reject the new LSP tunnel.

A similar situation can arise when one wants to increase the bandwidth of a TE tunnel. The new reservation will be for the full amount needed, but the actual allocation needed is only the delta between the new and old bandwidth. If policy is being applied to PATH messages by intermediate nodes, then a PATH message requesting too much bandwidth will be rejected. In this situation simply increasing the bandwidth request without changing the SENDER_TEMPLATE, could result in a tunnel being torn down, depending upon local policy.

The combination of the LSP_TUNNEL SESSION object and the SE reservation style naturally accommodates smooth transitions in bandwidth and routing. The idea is that the old and new LSP tunnels share resources along links which they have in common. The LSP_TUNNEL SESSION object is used to narrow the scope of the RSVP session to the particular TE tunnel in question. To uniquely identify a TE tunnel,

we use the combination of the destination IP address (an address of the node which is the egress of the tunnel), a Tunnel ID, and the tunnel ingress node's IP address, which is placed in the Extended Tunnel ID field.

During the reroute or bandwidth-increase operation, the tunnel ingress needs to appear as two different senders to the RSVP session. This is achieved by the inclusion of the "LSP ID", which is carried in the SENDER_TEMPLATE and FILTER_SPEC objects. Since the semantics of these objects are changed, a new C-Types are assigned.

To effect a reroute, the ingress node picks a new LSP ID and forms a new SENDER_TEMPLATE. The ingress node then creates a new ERO to define the new path. Thereafter the node sends a new Path Message using the original SESSION object and the new SENDER_TEMPLATE and ERO. It continues to use the old LSP and refresh the old Path message. On links that are not held in common, the new Path message is treated as a conventional new LSP tunnel setup. On links held in common, the shared SESSION object and SE style allow the LSP to be established sharing resources with the old LSP. Once the ingress node receives a Resv message for the new LSP, it can transition traffic to it and tear down the old LSP.

To effect a bandwidth-increase, a new Path Message with a new

LSP_ID
can be used to attempt a larger bandwidth reservation while the current LSP_ID continues to be refreshed to ensure that the reservation is not lost if the larger reservation fails.

2.6. Path MTU

Standard RSVP [1] and Int-Serv [11] provide the RSVP sender with the minimum MTU available between the sender and the receiver. This path MTU identification capability is also provided for LSPs established via RSVP.

Path MTU information is carried, depending on which is present, in the Integrated Services or Class-of-Service objects. When using Integrated Services objects, path MTU is provided based on the procedures defined in [11]. Path MTU identification when using Class-of-Service objects is defined in Section 4.8.

With standard RSVP, the path MTU information is used by the sender to check which IP packets exceed the path MTU. For packets that exceed the path MTU, the sender either fragments the packets or, when the IP datagram has the "Don't Fragment" bit set, issues an ICMP destination unreachable message. This path MTU related handling is also required for LSPs established via RSVP.

The following algorithm applies to all unlabeled IP datagrams and to any labeled packets which the node knows to be IP datagrams, to which labels need to be added before forwarding. For labeled packets the bottom of stack is found, the IP header examined.

Using the terminology defined in [5], an LSR MUST execute the following algorithm:

1. Let N be the number of bytes in the label stack (i.e, 4 times the number of label stack entries) including labels to be added by this node.

2. Let M be the smaller of the "Maximum Initially Labeled IP Datagram Size" or of (Path MTU - N).

When the size of the datagram (without labels) exceeds the value of M,

If the DF bit is not set in the IP header, then

(a) the datagram MUST be broken into fragments, each of whose size is no greater than M, and

(b) each fragment MUST be labeled and then forwarded.

If the DF bit is set in the IP header, then

(a) the datagram MUST NOT be forwarded

(b) Create an ICMP Destination Unreachable Message:
i. set its Code field [12] to "Fragmentation Required and DF Set",
ii. set its Next-Hop MTU field [13] to M

(c) If possible, transmit the ICMP Destination Unreachable Message to the source of the of the discarded datagram.

3. LSP Tunnel related Message Formats

Five new objects are defined in this section:

Object name	Applicable RSVP messages
LABEL_REQUEST	Path
LABEL	Resv
EXPLICIT_ROUTE	Path

RECORD_ROUTE	Path, Resv
SESSION_ATTRIBUTE	Path

New C-Types are also assigned for the SESSION, SENDER_TEMPLATE, FILTER_SPEC, FLOWSPEC objects.

Detailed descriptions of the new objects are given in later sections.

All new objects are OPTIONAL with respect to RSVP. An implementation can choose to support a subset of objects. However, the LABEL_REQUEST and LABEL objects are mandatory with respect to this specification.

The LABEL and RECORD_ROUTE objects, are sender specific. In Resv messages they MUST appear after the associated FILTER_SPEC and prior to any subsequent FILTER_SPEC.

The relative placement of EXPLICIT_ROUTE, LABEL_REQUEST, and SESSION_ATTRIBUTE objects is simply a recommendation. The ordering of these objects is not important, so an implementation MUST be prepared to accept objects in any order.

3.1. Path Message

The format of the Path message is as follows:

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

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

3.2. Resv Message

The format of the Resv message is as follows:


```
      <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> [ <RECORD_ROUTE> ]
                              | <FF flow descriptor list>
                              <FF flow descriptor>

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

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

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

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

      Note: LABEL and RECORD_ROUTE (if present), are bound to the
             preceding FILTER_SPEC. No more than one LABEL and/or
             RECORD_ROUTE may follow each FILTER_SPEC.

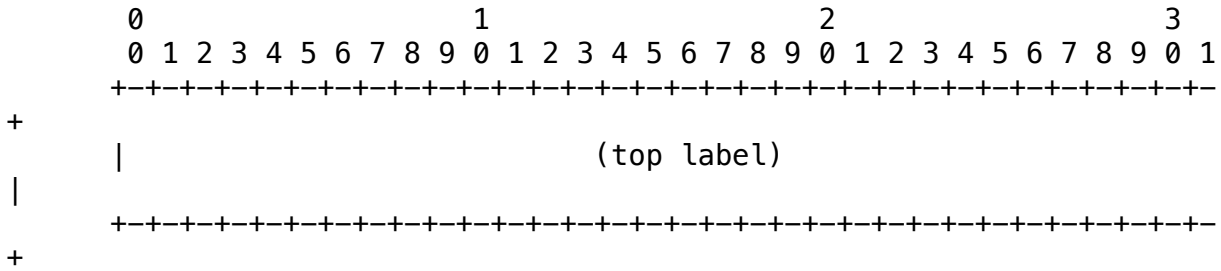
4. LSP Tunnel related Objects

4.1. Label Object

Labels MAY be carried in Resv messages. For the FF and SE styles, a label is associated with each sender. The label for a sender MUST immediately follow the FILTER_SPEC for that sender in the Resv message.

The LABEL object has the following format:

LABEL class = 16, C_Type = 1



The contents of a LABEL is a single label, encoded in 4 octets. Each generic MPLS label is an unsigned integer in the range 0 through 1048575. Generic MPLS labels and FR labels are encoded right aligned in 4 octets. ATM labels are encoded with the VPI right justified in bits 0-15 and the VCI right justified in bits 16-31.

4.1.1. Handling Label Objects in Resv messages

In MPLS a node may support multiple label spaces, perhaps associating a unique space with each incoming interface. For the purposes of the following discussion, the term "same label" means the identical label value drawn from the identical label space. Further, the following applies only to unicast sessions.

Labels received in Resv messages on different interfaces are always considered to be different even if the label value is the same.

4.1.1.1. Downstream

The downstream node selects a label to represent the flow. If a label range has been specified in the label request, the label MUST be drawn from that range. If no label is available the node sends a PathErr message with an error code of "routing problem" and an error value of "label allocation failure".

If a node receives a Resv message that has assigned the same label value to multiple senders, then that node MAY also assign a single value to those same senders or to any subset of those senders. Note that if a node intends to police individual senders to a session, it MUST assign unique labels to those senders.

In the case of ATM, one further condition applies. Some ATM nodes are not capable of merging streams. These nodes MAY indicate this by setting a bit in the label request to zero. The M-bit in the LABEL_REQUEST object of C-Type 2, label request with ATM label range, serves this purpose. The M-bit SHOULD be set by nodes which are merge capable. If for any senders the M-bit is not set, the downstream node MUST assign unique labels to those senders.

Once a label is allocated, the node formats a new LABEL object. The node then sends the new LABEL object as part of the Resv message to the previous hop. The LABEL object SHOULD be kept in the Reservation State Block. It is then used in the next Resv refresh event for formatting the Resv message.

A node is expected to send a Resv message before its refresh timers expire if the contents of the LABEL object change.

4.1.1.2. Upstream

A node uses the label carried in the LABEL object as the outgoing label associated with the sender. The router allocates a new label and binds it to the incoming interface of this session/sender. This is the same interface that the router uses to forward Resv messages to the previous hops.

Several circumstance can lead to an unacceptable label.

1. the node is a merge incapable ATM switch but the downstream node has assigned the same label to two senders
2. The implicit null label was assigned, but the node is not capable of doing a penultimate pop for the associated L3PID
3. The assigned label is outside the requested label range

In any of these events the node send a ResvErr message with an error code of "routing problem" and an error value of "unacceptable label value".

4.1.2. Non-support of the Label Object

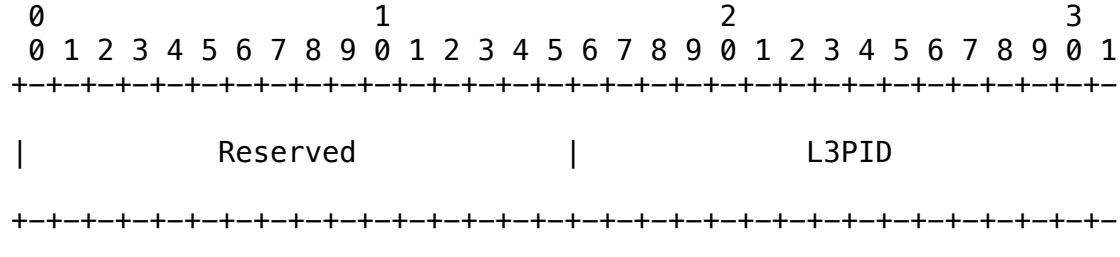
Under normal circumstances, a node should never receive a LABEL object in a Resv message unless it had included a LABEL_REQUEST object in the corresponding Path message. However, an RSVP router that does not recognize the LABEL object sends a ResvErr with the error code "Unknown object class" toward the receiver. This causes the reservation to fail.

4.2. Label Request Object

The Label Request Class is 19. Currently there three possible C_Types. Type 1 is a Label Request without label range. Type 2 is a label request with an ATM label range. Type 3 is a label request with a Frame Relay label range. The LABEL_REQUEST object formats are shown below.

4.2.1. Label Request without Label Range

Class = 19, C_Type = 1



Reserved

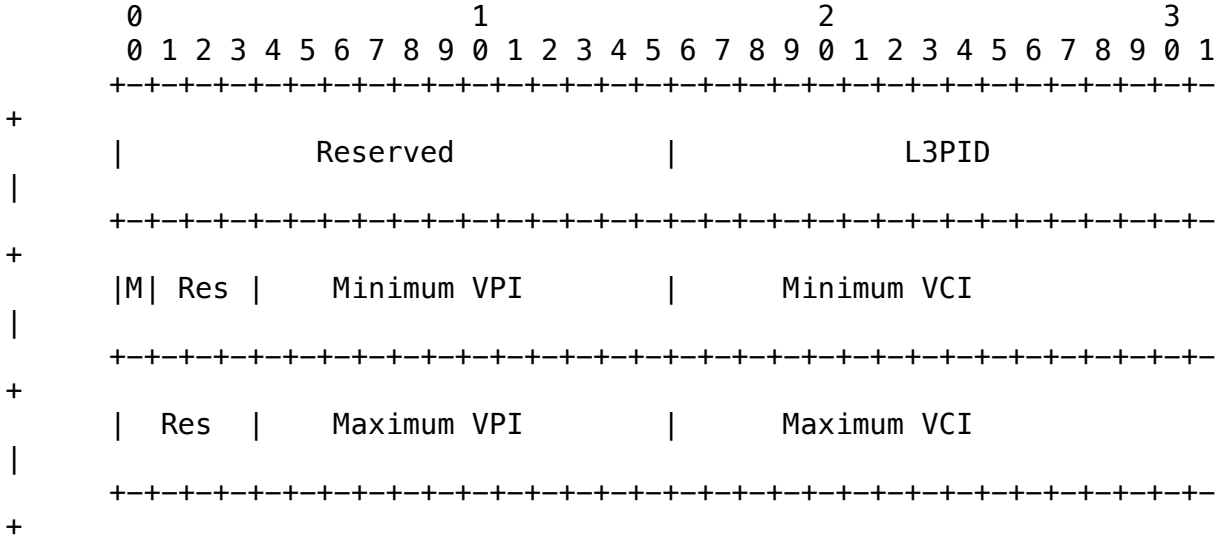
This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

L3PID

   an identifier of the layer 3 protocol using this path.
   Standard Ethertype values are used.

4.2.2. Label Request with ATM Label Range

Class = 19, C_Type = 2



Reserved (Res)

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

L3PID

   an identifier of the layer 3 protocol using this path.
   Standard Ethertype values are used.

M

Setting this bit to one indicates that the node is capable of merging in the data plane

Minimum VPI (12 bits)

This 12 bit field specifies the lower bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Minimum VCI (16 bits)

This 16 bit field specifies the lower bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VPI (12 bits)

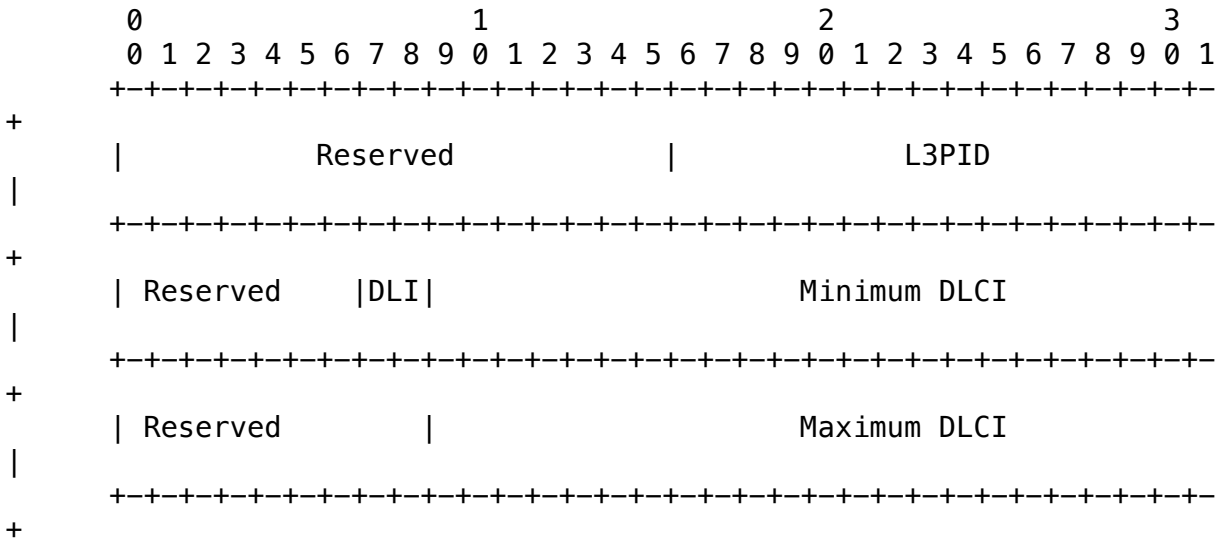
This 12 bit field specifies the upper bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VCI (16 bits)

This 16 bit field specifies the upper bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

4.2.3. Label Request with Frame Relay Label Range

Class = 19, C_Type = 3



Reserved

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

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

DLI

DLCI Length Indicator. The number of bits in the DLCI. The following values are supported:

Len	DLCI bits
0	10
2	23

Minimum DLCI

This 23-bit field specifies the lower bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the

originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

Maximum DLCI

This 23-bit field specifies the upper bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

4.2.4. Handling of LABEL_REQUEST

To establish an LSP tunnel the sender creates a Path message with a LABEL_REQUEST object. The LABEL_REQUEST object indicates that a label binding for this path is requested and provides an indication of the network layer protocol that is to be carried over this path. This permits non-IP network layer protocols to be sent down an LSP. This information can also be useful in actual label allocation, because some reserved labels are protocol specific, see [5].

The LABEL_REQUEST SHOULD be stored in the Path State Block, so that Path refresh messages will also contain the LABEL_REQUEST object. When the Path message reaches the receiver, the presence of the LABEL_REQUEST object triggers the receiver to allocate a label and to place the label in the LABEL object for the corresponding Resv message. If a label range was specified, the label MUST be allocated from that range. A receiver that accepts a LABEL_REQUEST object MUST include a LABEL object in Resv messages pertaining to that Path message. If a LABEL_REQUEST object was not present in the Path message, a node MUST NOT include a LABEL object in a Resv message for that Path message's session and PHOP.

A node that sends a LABEL_REQUEST object MUST be ready to accept

and
   correctly process a LABEL object in the corresponding Resv
messages.

   A node that recognizes a LABEL_REQUEST object, but that is unable
to
   support it (possibly because of a failure to allocate labels)
SHOULD
   send a PathErr with the error code "Routing problem" and the error
   value "MPLS label allocation failure." This includes the case
where
   a label range has been specified and a label cannot be allocated
from
   that range.

   A node which receives and forwards a Path message each with a
   LABEL_REQUEST object, MUST copy the L3PID from the received
   LABEL_REQUEST object to the forwarded LABEL_REQUEST object.

   If the receiver cannot support the protocol L3PID, it SHOULD send a
   PathErr with the error code "Routing problem" and the error value
   "Unsupported L3PID." This causes the RSVP session to fail.

4.2.5. Non-support of the Label Request Object

   An RSVP router that does not recognize the LABEL_REQUEST object
sends
   a PathErr with the error code "Unknown object class" toward the
   sender. An RSVP router that recognizes the LABEL_REQUEST object
but
   does not recognize the C_Type sends a PathErr with the error code
   "Unknown object C_Type" toward the sender. This causes the path

   setup to fail. The sender should notify management that a LSP
cannot
   be established and possibly take action to continue the reservation
   without the LABEL_REQUEST.

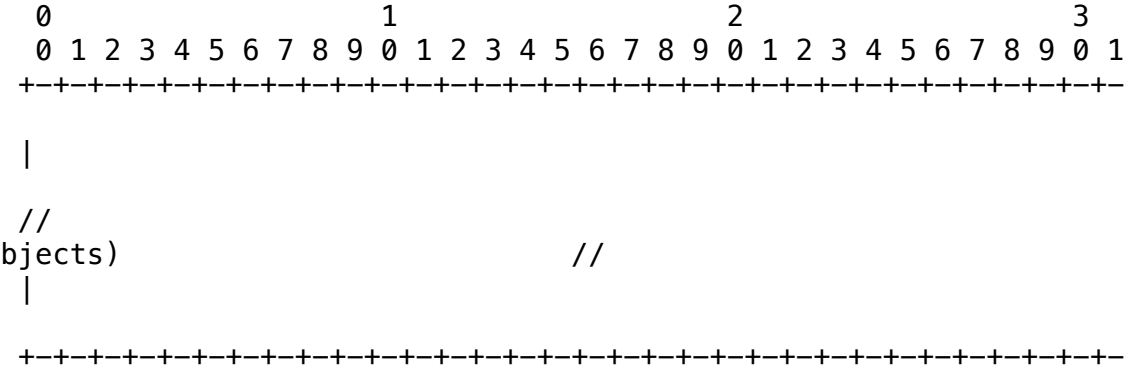
   RSVP is designed to cope gracefully with non-RSVP routers anywhere

between senders and receivers. However, obviously, non-RSVP routers cannot convey labels via RSVP. This means that if a router has a neighbor that is known to not be RSVP capable, the router MUST NOT advertise the LABEL_REQUEST object when sending messages that pass through the non-RSVP routers. The router SHOULD send a PathErr back to the sender, with the error code "Routing problem" and the error value "MPLS being negotiated, but a non-RSVP capable router stands in the path." This same message SHOULD be sent, if a router receives a LABEL_REQUEST object in a message from a non-RSVP capable router. See [1] for a description of how a downstream router can determine the presence of non-RSVP routers.

4.3. Explicit Route Object

Explicit routes are specified via the EXPLICIT_ROUTE object (ERO). The Explicit Route Class is 20. Currently one C_Type is defined, Type 1 Explicit Route. The EXPLICIT_ROUTE object has the following format:

Class = 20, C_Type = 1



Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. The subobjects are defined in section 4.3.3 below.

If a Path message contains multiple EXPLICIT_ROUTE objects, only the first object is meaningful. Subsequent EXPLICIT_ROUTE objects MAY be ignored and SHOULD NOT be propagated.

4.3.1. Applicability

The EXPLICIT_ROUTE object is intended to be used only for unicast situations. Applications of explicit routing to multicast are a topic for further research.

The EXPLICIT_ROUTE object is to be used only when all routers along the explicit route support RSVP and the EXPLICIT_ROUTE object. The EXPLICIT_ROUTE object is assigned a class value of the form 0bbbbbbb.
RSVP routers that do not support the object will therefore respond with an "Unknown Object Class" error.

4.3.2. Semantics of the Explicit Route Object

An explicit route is a particular path in the network topology. Typically, the explicit route is determined by a node, with the intent of directing traffic along that path.

An explicit route is described as a list of groups of nodes along the explicit route. In addition to the ability to identify specific nodes along the path, an explicit route can identify a group of nodes that must be traversed along the path. This capability allows the routing system a significant amount of local flexibility in fulfilling a request for an explicit route. This capability allows the generator of the explicit route to have imperfect information about the details of the path.

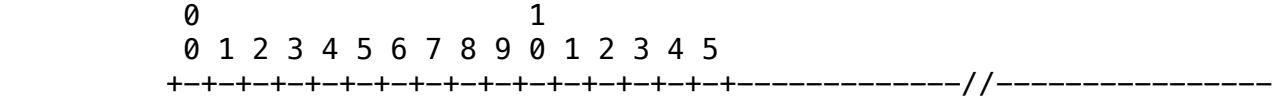
The explicit route is encoded as a series of subobjects contained in an EXPLICIT_ROUTE object. Each subobject identifies a group of nodes in the explicit route. An explicit route is thus a specification of groups of nodes to be traversed.

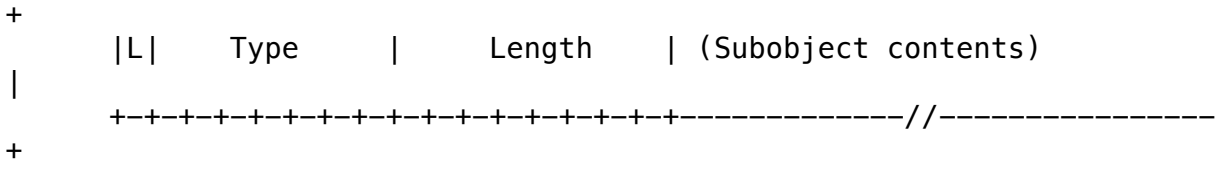
To formalize the discussion, we call each group of nodes an abstract node. Thus, we say that an explicit route is a specification of a set of abstract nodes to be traversed. If an abstract node consists of only one node, we refer to it as a simple abstract node.

As an example of the concept of abstract nodes, consider an explicit route that consists solely of Autonomous System number subobjects. Each subobject corresponds to an Autonomous System in the global topology. In this case, each Autonomous System is an abstract node, and the explicit route is a path that includes each of the specified Autonomous Systems. There may be multiple hops within each Autonomous System, but these are opaque to the source node for the explicit route.

4.3.3. Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. Each subobject has the form:





L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

The Type indicates the type of contents of the subobject. Currently defined values are:

0	Reserved
1	IPv4 prefix
2	IPv6 prefix
32	Autonomous system number

Length

The Length contains the total length of the subobject in bytes, including the L, Type and Length fields. The Length MUST be at least 4, and MUST be a multiple of 4.

4.3.3.1. Strict and Loose Subobjects

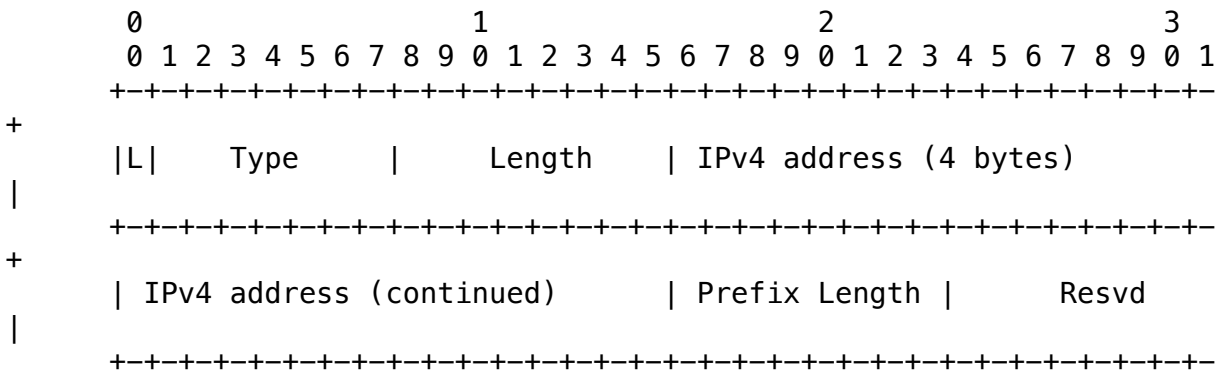
The L bit in the subobject is a one-bit attribute. If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.' For brevity, we say that if the value of the subobject attribute is 'loose' then it is a 'loose subobject.' Otherwise, it's a 'strict subobject.' Further, we say that the abstract node of a strict or loose subobject is a strict or a loose node, respectively. Loose and strict nodes are always interpreted relative to their prior abstract nodes.

The path between a strict node and its preceding node MUST include

only network nodes from the strict node and its preceding abstract node.

The path between a loose node and its preceding node MAY include other network nodes that are not part of the strict node or its preceding abstract node.

4.3.3.2. Subobject 1: IPv4 prefix



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

An IPv4 address. This address is treated as a prefix based on the prefix length value below. Bits beyond the prefix are ignored on receipt and SHOULD be set to zero on transmission.

Prefix length

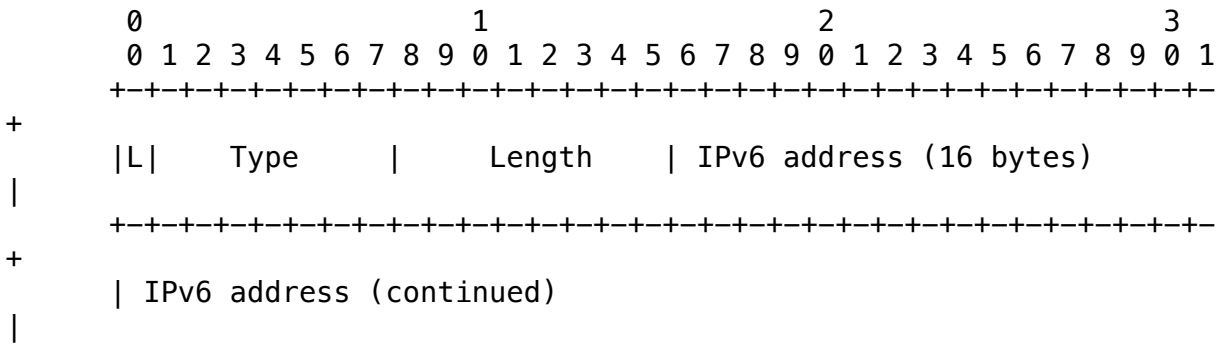
Length in bits of the IPv4 prefix

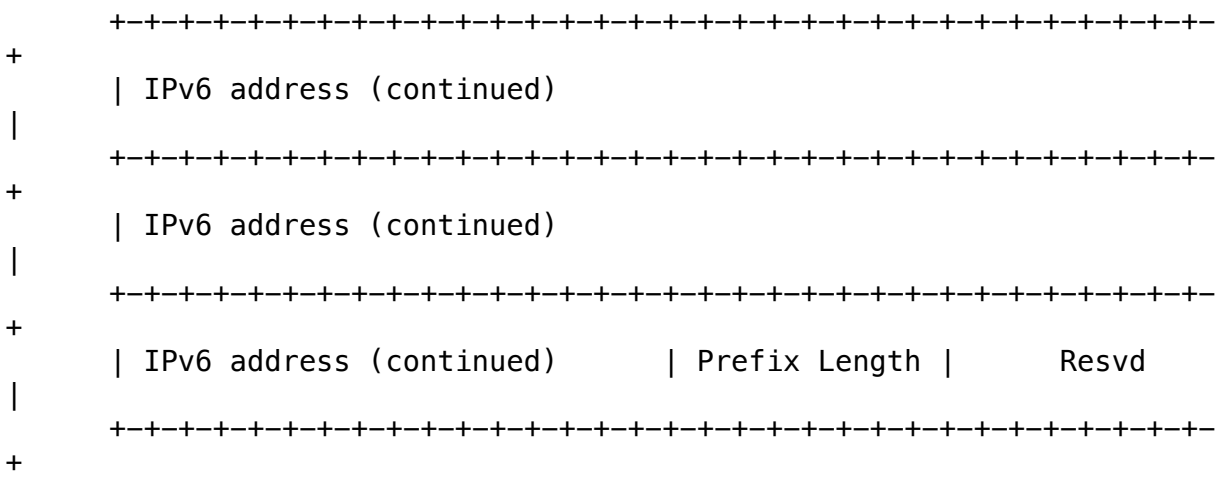
Padding

Zero on transmission. Ignored on receipt.

The contents of an IPv4 prefix subobject are a 4-octet IPv4 address, a 1-octet prefix length, and a 1-octet pad. The abstract node represented by this subobject is the set of nodes that have an IP address which lies within this prefix. Note that a prefix length of 32 indicates a single IPv4 node.

4.3.3.3. Subobject 2: IPv6 Prefix





L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

0x02 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

An IPv6 address. This address is treated as a prefix based on the prefix length value below. Bits beyond the prefix are ignored on receipt and SHOULD be set to zero on transmission.

Prefix Length

Length in bits of the IPv6 prefix.

Padding

Zero on transmission. Ignored on receipt.

The contents of an IPv6 prefix subobject are a 16-octet IPv6 address, a 1-octet prefix length, and a 1-octet pad. The abstract node represented by this subobject is the set of nodes that have an IP address which lies within this prefix. Note that a prefix length of 128 indicates a single IPv6 node.

4.3.3.4. Subobject 32: Autonomous System Number

The contents of an Autonomous System (AS) number subobject are a 2-octet AS number. The abstract node represented by this subobject is the set of nodes belonging to the autonomous system.

The length of the AS number subobject is 4 octets.

4.3.4. Processing of the Explicit Route Object

4.3.4.1. Selection of the Next Hop

A node receiving a Path message containing an EXPLICIT_ROUTE object must determine the next hop for this path. This is necessary because the next abstract node along the explicit route might be an IP subnet or an Autonomous System. Therefore, selection of this next hop may involve a decision from a set of feasible alternatives. The criteria used to make a selection from feasible alternatives is implementation dependent and can also be impacted by local policy, and is beyond the scope of this specification. However, it is assumed that each node

will make a best effort attempt to determine a loop-free path. Note
that paths so determined can be overridden by local policy.

To determine the next hop for the path, a node performs the following
steps:

1) The node receiving the RSVP message MUST first evaluate the first
subobject. If the node is not part of the abstract node described by
the first subobject, it has received the message in error and SHOULD
return a "Bad initial subobject" error. If there is no first
subobject, the message is also in error and the system SHOULD return

a "Bad EXPLICIT_ROUTE object" error.

2) If there is no second subobject, this indicates the end of the
explicit route. The EXPLICIT_ROUTE object SHOULD be removed from the
Path message. This node may or may not be the end of the path.
Processing continues with section 4.3.4.2, where a new EXPLICIT_ROUTE
object MAY be added to the Path message.

3) Next, the node evaluates the second subobject. If the node is
also a part of the abstract node described by the second subobject,
then the node deletes the first subobject and continues processing
with step 2, above. Note that this makes the second subobject into
the first subobject of the next iteration and allows the node to
identify the next abstract node on the path of the message after
possible repeated application(s) of steps 2 and 3.

4) Abstract Node Border Case: The node determines whether it is
topologically adjacent to the abstract node described by the second
subobject. If so, the node selects a particular next hop which is a

member of the abstract node. The node then deletes the first subobject and continues processing with section 4.3.4.2.

5) Interior of the Abstract Node Case: Otherwise, the node selects a next hop within the abstract node of the first subobject (which the node belongs to) that is along the path to the abstract node of the second subobject (which is the next abstract node). If no such path exists then there are two cases:

5a) If the second subobject is a strict subobject, there is an error and the node SHOULD return a "Bad strict node" error.

5b) Otherwise, if the second subobject is a loose subobject, the node selects any next hop that is along the path to the next abstract node. If no path exists, there is an error, and the node SHOULD return a "Bad loose node" error.

6) Finally, the node replaces the first subobject with any subobject that denotes an abstract node containing the next hop. This is necessary so that when the explicit route is received by the next hop, it will be accepted.

4.3.4.2. Adding subobjects to the Explicit Route Object

After selecting a next hop, the node MAY alter the explicit route in the following ways.

If, as part of executing the algorithm in section 4.3.4.1, the

EXPLICIT_ROUTE object is removed, the node MAY add a new EXPLICIT_ROUTE object.

Otherwise, if the node is a member of the abstract node for the

first
   subobject, a series of subobjects MAY be inserted before the first
   subobject or MAY replace the first subobject. Each subobject in
this
   series MUST denote an abstract node that is a subset of the current
   abstract node.

   Alternately, if the first subobject is a loose subobject, an
   arbitrary series of subobjects MAY be inserted prior to the first
   subobject.

4.3.5. Loops

   While the EXPLICIT_ROUTE object is of finite length, the existence
of
   loose nodes implies that it is possible to construct forwarding
loops
   during transients in the underlying routing protocol. This can be
   detected by the originator of the explicit route through the use of
   another opaque route object called the RECORD_ROUTE object. The
   RECORD_ROUTE object is used to collect detailed path information
and
   is useful for loop detection and for diagnostics.

4.3.6. Forward Compatibility

   It is anticipated that new subobjects may be defined over time. A
   node which encounters an unrecognized subobject during its normal
ERO
   processing sends a PathErr with the error code "Routing Error" and
   error value of "Bad Explicit Route Object" toward the sender. The
   EXPLICIT_ROUTE object is included, truncated (on the left) to the
   offending subobject. The presence of an unrecognized subobject
which
   is not encountered in a node's ERO processing SHOULD be ignored.
It
   is passed forward along with the rest of the remaining ERO stack.

4.3.7. Non-support of the Explicit Route Object

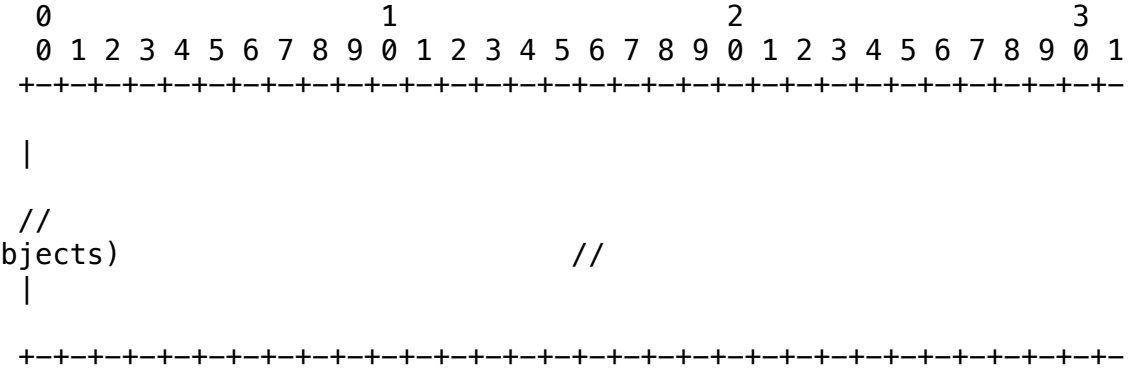
   An RSVP router that does not recognize the EXPLICIT_ROUTE object
   sends a PathErr with the error code "Unknown object class" toward
the
   sender. This causes the path setup to fail. The sender should
   notify management that a LSP cannot be established and possibly
take
   action to continue the reservation without the EXPLICIT_ROUTE or

via
   a different explicit route.

4.4. Record Route Object

   Routes can be recorded via the RECORD_ROUTE object (RRO).
   Optionally, labels may also be recorded. The Record Route Class is
   21. Currently one C_Type is defined, Type 1 Record Route. The
   RECORD_ROUTE object has the following format:

      Class = 21, C_Type = 1



      Subobjects

         The contents of a RECORD_ROUTE object are a series of
         variable-length data items called subobjects. The subobjects
         are defined in section 4.4.1 below.

The RRO can be present in both RSVP Path and Resv messages. If a Path message contains multiple RROs, only the first RRO is meaningful. Subsequent RROs SHOULD be ignored and SHOULD NOT be propagated. Similarly, if in a Resv message multiple RROs are encountered following a FILTER_SPEC before another FILTER_SPEC is encountered, only the first RRO is meaningful. Subsequent RROs SHOULD be ignored and SHOULD NOT be propagated.

4.4.1. Subobjects

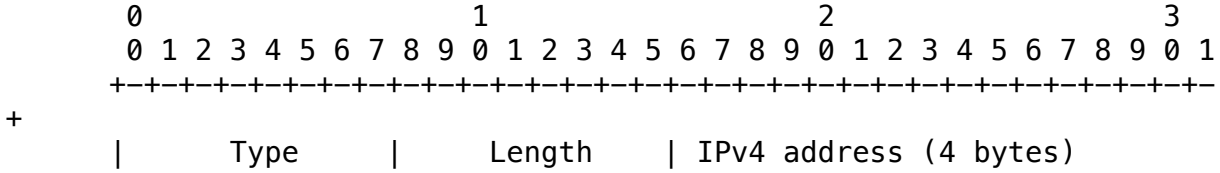
The contents of a RECORD_ROUTE object are a series of variable-length data items called subobjects. Each subobject has its own Length field. The length contains the total length of the subobject in bytes, including the Type and Length fields. The length MUST always be a multiple of 4, and at least 4.

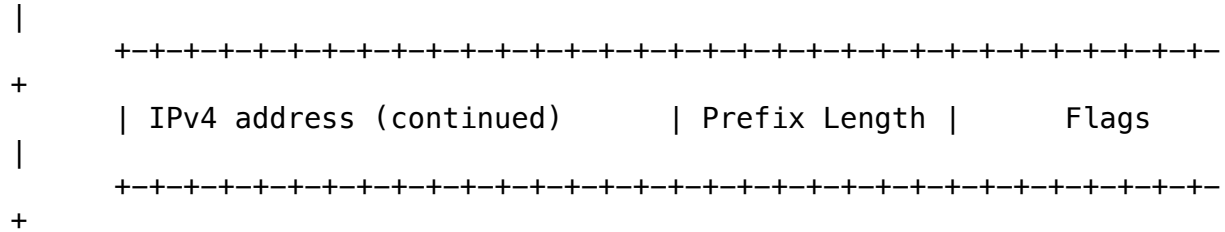
Subobjects are organized as a last-in-first-out stack. The first subobject relative to the beginning of RRO is considered the top. The last subobject is considered the bottom. When a new subobject is added, it is always added to the top.

An empty RRO with no subobjects is considered illegal.

Three kinds of subobjects are currently defined.

4.4.1.1. Subobject 1: IPv4 address





Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

A 32-bit unicast, host address. Any network-reachable interface address is allowed here. Illegal addresses, such as certain loopback addresses, SHOULD NOT be used.

Prefix length

32

Flags

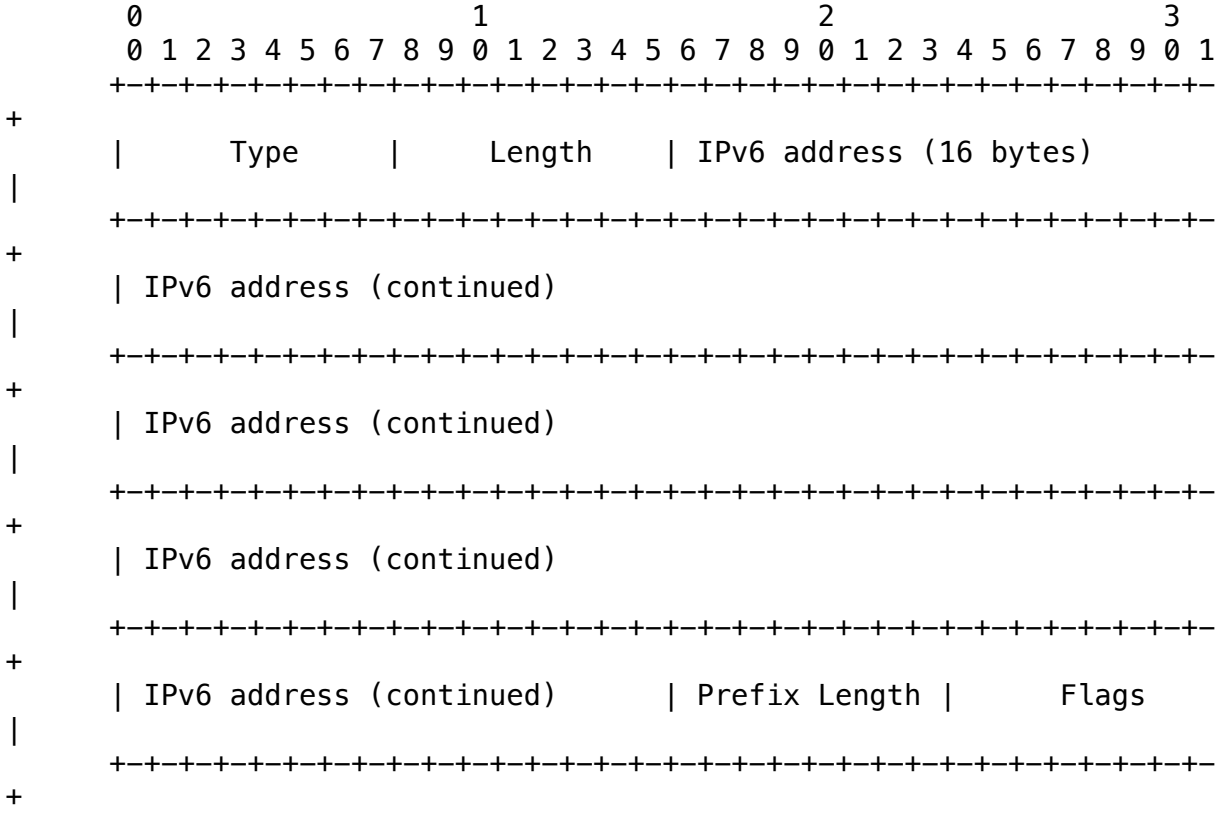
0x01 Local protection available

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).

4.4.1.2. Subobject 2: IPv6 address



Type

0x02 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

A 128-bit unicast host address.

Prefix length

128

Flags

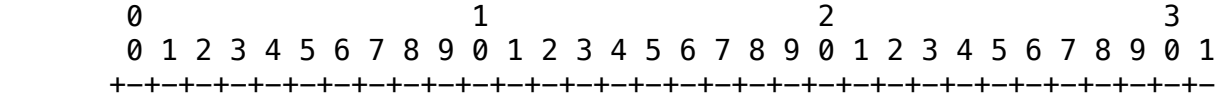
0x01 Local protection available

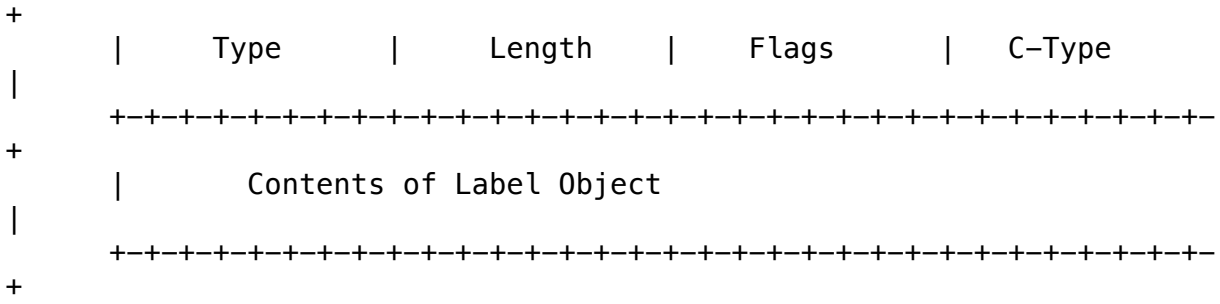
Indicates that the link downstream of this node is protected via a local repair mechanism. This flag can only be set if the Local protection flag was set in the SESSION_ATTRIBUTE object of the cooresponding Path nessage.

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).

4.4.1.3. Subobject 0x03, Label





Type

0x03 Label

Length

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

Flags

0x01 = Global label
This flag indicates that the label will be understood if received on any interface.

C-Type

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

Contents of Label Object

The contents of the Label Object. Copied from the Label Object

4.4.2. Applicability

Only the procedures for use in unicast sessions are defined here.

There are three possible uses of RRO in RSVP. First, an RRO can function as a loop detection mechanism to discover L3 routing loops, or loops inherent in the explicit route. The exact procedure for doing so is described later in this document.

Second, an RRO collects up-to-date detailed path information hop-by-hop about RSVP sessions, providing valuable information to the sender or receiver. Any path change (due to network topology changes) will be reported.

Third, RRO syntax is designed so that, with minor changes, the whole object can be used as input to the EXPLICIT_ROUTE object. This is useful if the sender receives RRO from the receiver in a Resv message, applies it to EXPLICIT_ROUTE object in the next Path message in order to "pin down session path".

4.4.3. Processing RRO

Typically, a node initiates an RSVP session by adding the RRO to the Path message. The initial RRO contains only one subobject - the sender's IP addresses. If the node also desires label recording, it sets the Label_Recording flag in the SESSION_ATTRIBUTE object.

When a Path message containing an RRO is received by an intermediate router, the router stores a copy of it in the Path State Block. The RRO is then used in the next Path refresh event for formatting Path messages. When a new Path message is to be sent, the router adds a new subobject to the RRO and appends the resulting RRO to the Path message before transmission.

The newly added subobject MUST be this router's IP address. The

address to be added SHOULD be the interface address of the outgoing Path messages. If there are multiple addresses to choose from, the decision is a local matter. However, it is RECOMMENDED that the same address be chosen consistently.

When the Label_Recording flag is set in the SESSION_ATTRIBUTE object, nodes doing route recording SHOULD include a Label Record subobject. If the node is using a global label space, then it SHOULD set the Global Label flag.

The Label Record subobject is pushed onto the RECORD_ROUTE object prior to pushing on the node's IP address. A node MUST NOT push on a Label Record subobject without also pushing on an IPv4 or IPv6 subobject.

Note that on receipt of the initial Path message, a node is unlikely to have a label to include. Once a label is obtained, the node SHOULD include the label in the RRO in the next Path refresh event.

If the newly added subobject causes the RRO to be too big to fit in a Path (or Resv) message, the RRO object SHALL be dropped from the message and message processing continues as normal. A PathErr (or ResvErr) message SHOULD be sent back to the sender (or receiver). An error code of "Notify" and an error value of "RRO too large for MTU" is used. If the receiver receives such a ResvErr, it SHOULD send a PathErr message with error code of "Notify" and an error value of "RRO notification".

A sender receiving either of these error values SHOULD remove the RRO from the Path message.

Nodes SHOULD resend the above PathErr or ResvErr message each n seconds where n is the greater of 15 and the refresh interval for the

associated Path or RESV message. The node MAY apply limits and/or back-off timers to limit the number of messages sent.

An RSVP router can decide to send Path messages before its refresh time if the RRO in the next Path message is different from the previous one. This can happen if the contents of the RRO received from the previous hop router changes or if this RRO is newly added to (or deleted from) the Path message.

When the destination node of an RSVP session receives a Path message with an RRO, this indicates that the sender node needs route recording. The destination node initiates the RRO process by adding an RRO to Resv messages. The processing mirrors that of the Path messages. The only difference is that the RRO in a Resv message records the path information in the reverse direction.

Note that each node along the path will now have the complete route from source to destination. The Path RRO will have the route from the source to this node; the Resv RRO will have the route from this node to the destination. This is useful for network management.

A received Path message without an RRO indicates that the sender node no longer needs route recording. Subsequent Resv messages SHALL NOT contain an RRO.

4.4.4. Loop Detection

As part of processing an incoming RRO, an intermediate router looks into all subobjects contained within the RRO. If the router determines that it is already in the list, a forwarding loop exists.

An RSVP session is loop-free if downstream nodes receive Path messages or upstream nodes receive Resv messages with no routing

loops detected in the contained RRO.

There are two broad classifications of forwarding loops. The first class is the transient loop, which occurs as a normal part of operations as L3 routing tries to converge on a consistent forwarding path for all destinations. The second class of forwarding loop is the permanent loop, which normally results from network mis-configuration.

The action performed by a node on receipt of an RRO depends on the message type in which the RRO is received.

For Path messages containing a forwarding loop, the router builds and sends a "Routing problem" PathErr message, with the error value "loop detected," and drops the Path message. Until the loop is eliminated, this session is not suitable for forwarding data packets. How the loop eliminated is beyond the scope of this document.

For Resv messages containing a forwarding loop, the router simply drops the message. Resv messages should not loop if Path messages do not loop.

4.4.5. Forward Compatibility

New subobjects may be defined for the RRO. When processing an RRO, unrecognized subobjects SHOULD be ignored and passed on. When processing an RRO for loop detection, a node SHOULD parse over any unrecognized objects. Loop detection works by detecting subobjects which were inserted by the node itself on an earlier pass of the

object. This ensures that the subobjects necessary for loop detection are always understood.

4.4.6. Non-support of RRO

The RRO object is to be used only when all routers along the path support RSVP and the RRO object. The RRO object is assigned a class value of the form 0bbbbbbb. RSVP routers that do not support the object will therefore respond with an "Unknown Object Class" error.

4.5. Error Codes for ERO and RRO

In the processing described above, certain errors must be reported as either a "Routing Problem" or "Notify". The value of the "Routing Problem" error code is 24; the value of the "Notify" error code is 25.

The following defines error values for the Routing Problem Error Code:

Value	Error:
1	Bad EXPLICIT_ROUTE object
2	Bad strict node
3	Bad loose node
4	Bad initial subobject
5	No route available toward destination
6	Unacceptable label value
7	RRO indicated routing loops
8	MPLS being negotiated, but a non-RSVP-capable router stands in the path
9	MPLS label allocation failure
10	Unsupported L3PID

For the Notify Error Code, the 16 bits of the Error Value field are:

```
        ss00 cccc cccc cccc
```

The high order bits are as defined under Error Code 1. (See [1]).

When ss = 00, the following subcodes are defined:

```
         1    RRO too large for MTU
         2    RRO notification
```

4.6. Session, Sender Template, and Filter Spec Objects

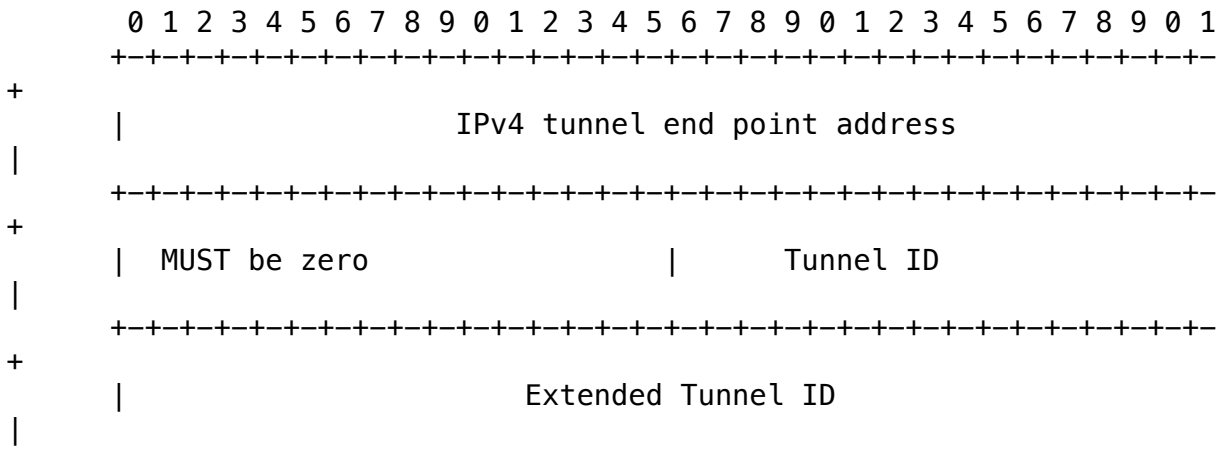
New C-Types are defined for the SESSION, SENDER_TEMPLATE and FILTER_SPEC objects.

The LSP_TUNNEL objects have the following format:

4.6.1. Session Object

4.6.1.1. LSP_TUNNEL_IPv4 Session Object

Class = SESSION, LSP_TUNNEL_IPv4 C-Type = 7



```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
+
```

IPv4 tunnel end point address

IPv4 address of the egress node for the tunnel.

Tunnel ID

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

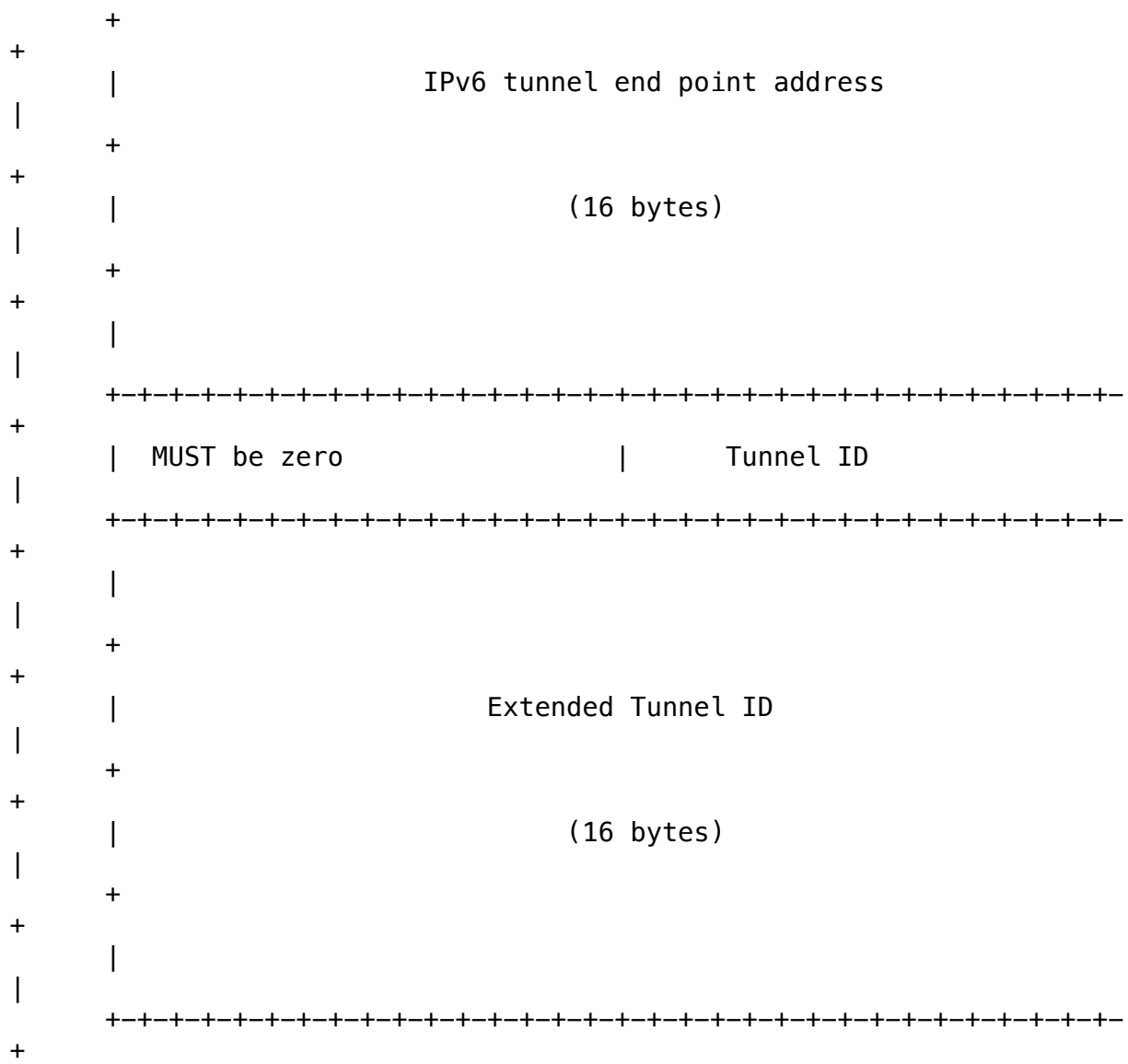
Extended Tunnel ID

A 32-bit identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv4 address here as a globally unique identifier.

4.6.1.2. LSP_TUNNEL_IPv6 Session Object

Class = SESSION, LSP_TUNNEL_IPv6 C_Type = 8

```
       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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
+
      |
|
```



IPv6 tunnel end point address

IPv6 address of the egress node for the tunnel.

Tunnel ID

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

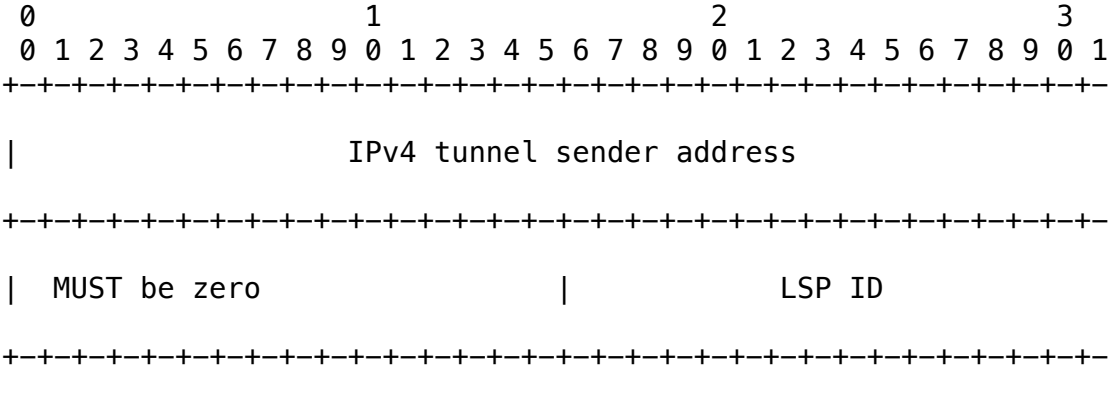
Extended Tunnel ID

A 16-byte identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv6 address here as a globally unique identifier.

4.6.2. Sender Template Object

4.6.2.1. LSP_TUNNEL_IPv4 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv4 C-Type = 7



IPv4 tunnel sender address

IPv4 address for a sender node

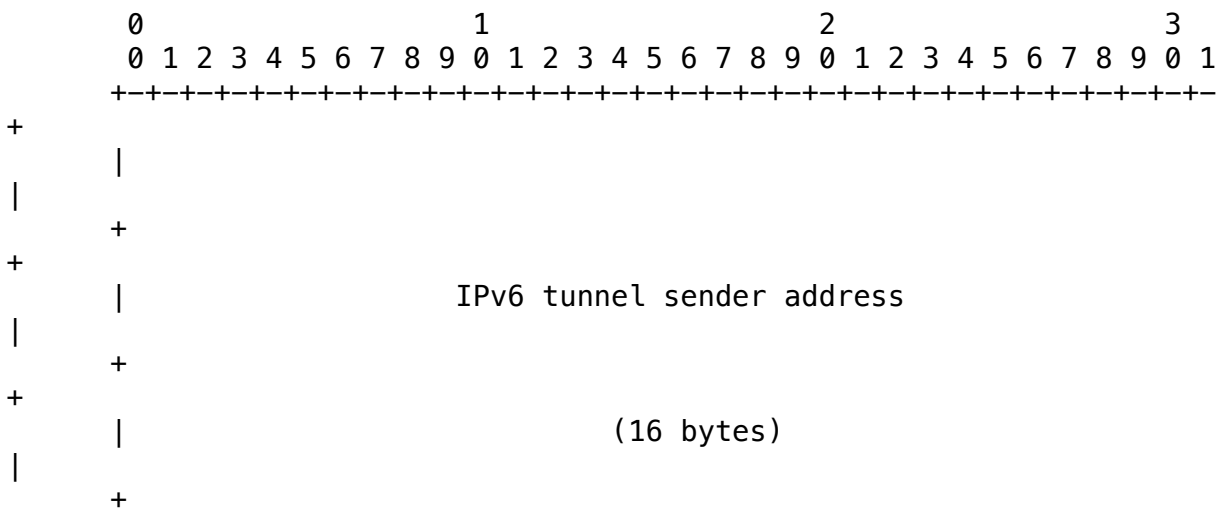
LSP ID

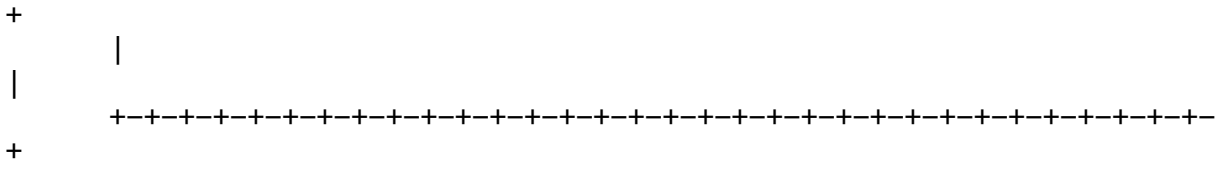
A 16-bit identifier used in the SENDER_TEMPLATE and the

FILTER_SPEC that can be changed to allow a sender to share
resources with itself.

4.6.2.2. LSP_TUNNEL_IPv6 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv6 C_Type = 8





IPv6 tunnel sender address

IPv6 address for a sender node

LSP ID

A 16-bit identifier used in the SENDER_TEMPLATE and the FILTER_SPEC that can be changed to allow a sender to share resources with itself.

4.6.3. Filter Specification Object

4.6.3.1. LSP_TUNNEL_IPv4 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv4 C-Type = 7

The format of the LSP_TUNNEL_IPv4 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv4 SENDER_TEMPLATE object.

4.6.3.2. LSP_TUNNEL_IPv6 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv6 C_Type = 8

The format of the LSP_TUNNEL_IPv6 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv6 SENDER_TEMPLATE object.

4.6.4. Reroute and Bandwidth Increase Procedure

This section describes how to setup a tunnel that is capable of maintaining resource reservations (without double counting) while it is being rerouted or while it is attempting to increase its bandwidth. In the initial Path message, the ingress node forms a SESSION object, assigns a Tunnel_ID, and places its IPv4 address in the Extended_Tunnel_ID It also forms a SENDER_TEMPLATE and assigns a LSP_ID. Tunnel setup then proceeds according to the normal procedure.

On receipt of the Path message, the egress node sends a Resv message with the STYLE Shared Explicit toward the ingress node.

When an ingress node with an established path wants to change that path, it forms a new Path message as follows. The existing SESSION object is used. In particular the Tunnel_ID and Extended_Tunnel_ID are unchanged. The ingress node picks a new LSP_ID to form a new SENDER_TEMPLATE. It creates an EXPLICIT_ROUTE object for the new route. The new Path message is sent. The ingress node refreshes both the old and new path messages

The egress node responds with a Resv message with an SE flow descriptor formatted as:

```
<FLOWSPEC><old_FILTER_SPEC><old_LABEL_OBJECT><new_FILTER_SPEC>
          <new_LABEL_OBJECT>
```

(Note that if the PHOPs are different, then two messages are sent each with the appropriate FILTER_SPEC and LABEL_OBJECT.)

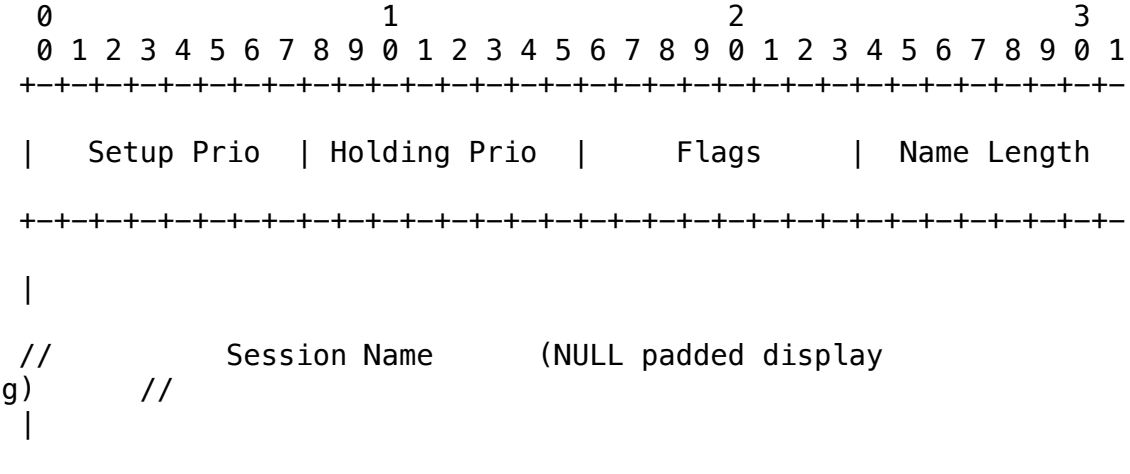
When the ingress node receives the Resv Message(s), it may begin using the new route. It SHOULD send a PathTear message for the old route.

4.7. Session Attribute Object

The Session Attribute Class is 207. Two C_Types are defined, LSP_TUNNEL, C-Type = 7 and LSP_TUNNEL_RA, C-Type = 1. The LSP_TUNNEL_RA C-Type includes all the same fields as the LSP_TUNNEL C-Type. Additionally it carries resource affinity information. The formats are as follows:

4.7.1. Format without resource affinities

SESSION_ATTRIBUTE class = 207, LSP_TUNNEL C-Type = 7



```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+
```

      Setup Priority

         The priority of the session with respect to taking
resources,
         in the range of 0 to 7. The value 0 is the highest
priority.
         The Setup Priority is used in deciding whether this session
can
         preempt another session.

      Holding Priority

         The priority of the session with respect to holding
resources,
         in the range of 0 to 7. The value 0 is the highest
priority.
         Holding Priority is used in deciding whether this session
can
         be preempted by another session.

      Flags

          0x01 Local protection desired

                This flag permits transit routers to use a local
repair

mechanism which may result in violation of the explicit
route object. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

0x02 Label recording desired

This flag indicates that label information should be included when doing a route record.

0x04 SE Style desired

This flag indicates that the tunnel ingress node may choose to reroute this tunnel without tearing it down. A tunnel egress node SHOULD use the SE Style when responding with a Resv message.

Name Length

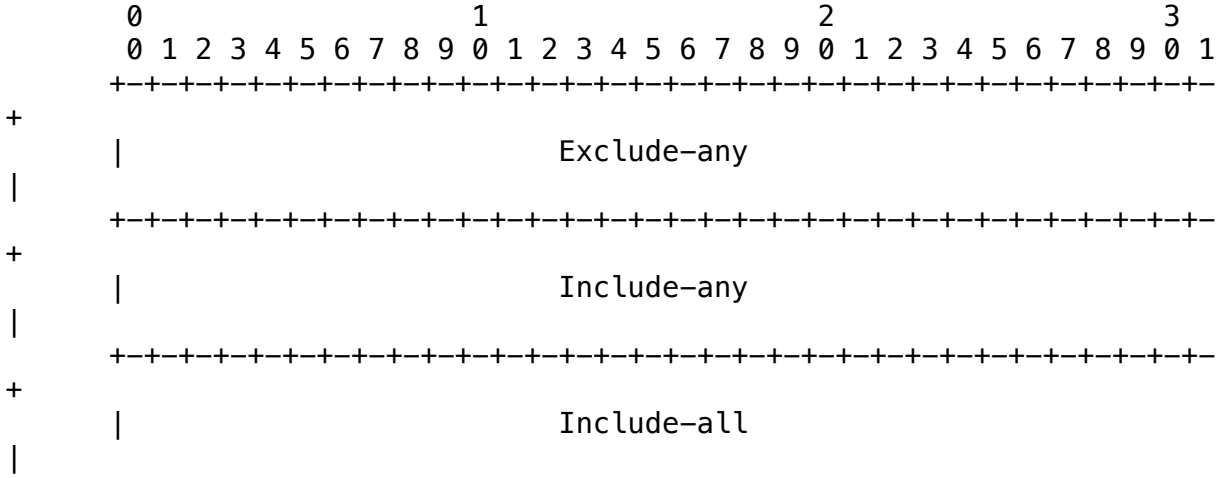
The length of the display string before padding, in bytes.

Session Name

A null padded string of characters.

4.7.2. Format with resource affinities

SESSION_ATTRIBUTE class = 207, LSP_TUNNEL_RA C-Type = 1



Exclude-any

A 32-bit vector representing a set of attribute filters associated with a tunnel any of which renders a link unacceptable.

Include-any

A 32-bit vector representing a set of attribute

filters
associated with a tunnel any of which renders a link acceptable
(with respect to this test). A null set (all bits set to zero)
automatically passes.

Include-all

A 32-bit vector representing a set of attribute filters
associated with a tunnel all of which must be present for a
link to be acceptable (with respect to this test). A null set
(all bits set to zero) automatically passes.

Setup Priority

The priority of the session with respect to taking resources,
in the range of 0 to 7. The value 0 is the highest priority.
The Setup Priority is used in deciding whether this session can
preempt another session.

Holding Priority

The priority of the session with respect to holding resources,
in the range of 0 to 7. The value 0 is the highest priority.
Holding Priority is used in deciding whether this session can
be preempted by another session.

Flags

0x01 Local protection desired

This flag permits transit routers to use a local repair mechanism which may result in violation of the explicit route object. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

0x02 Label recording desired

This flag indicates that label information should be included when doing a route record.

0x04 SE Style desired

This flag indicates that the tunnel ingress node may choose to reroute this tunnel without tearing it down. A tunnel egress node SHOULD use the SE Style when responding with a Resv message.

Name Length

The length of the display string before padding, in bytes.

Session Name

A null padded string of characters.

4.7.3. Procedures applying to both C-Types

The support of setup and holding priorities is OPTIONAL. A node can recognize this information but be unable to perform the requested operation. The node SHOULD pass the information downstream unchanged.

As noted above, preemption is implemented by two priorities. The Setup Priority is the priority for taking resources. The Holding

Priority is the priority for holding a resource. Specifically, the Holding Priority is the priority at which resources assigned to this session will be reserved. The Setup Priority SHOULD never be higher than the Holding Priority for a given session.

The setup and holding priorities are directly analogous to the preemption and defending priorities as defined in [9]. While the interaction of these two objects is ultimately a matter of policy, the following default interaction is RECOMMENDED.

When both objects are present, the preemption priority policy element is used. A mapping between the priority spaces is defined as follows. A session attribute priority S is mapped to a preemption priority P by the formula P = 2^(14-2S). The reverse mapping is shown in the following table.

Preemption Priority	Session Attribute Priority
0 - 3	7
4 - 15	6
16 - 63	5
64 - 255	4
256 - 1023	3
1024 - 4095	2
4096 - 16383	1
16384 - 65535	0

When a new Path message is considered for admission, the bandwidth requested is compared with the bandwidth available at the priority specified in the Setup Priority.

If the requested bandwidth is not available a PathErr message is returned with an Error Code of 01, Admission Control Failure, and an Error Value of 0x0002. The first 0 in the Error Value indicates a globally defined subcode and is not informational. The 002 indicates "requested bandwidth unavailable".

If the requested bandwidth is less than the unused bandwidth then processing is complete. If the requested bandwidth is available, but is in use by lower priority sessions, then lower priority sessions (beginning with the lowest priority) can be pre-empted to free the necessary bandwidth.

When pre-emption is supported, each pre-empted reservation triggers a
TC_Preempt() upcall to local clients, passing a subcode that
indicates the reason. A ResvErr and/or PathErr with the code "Policy
Control failure" SHOULD be sent toward the downstream receivers and
upstream senders.

The support of local-protection is OPTIONAL. A node may recognize
the local-protection Flag but may be unable to perform the requested
operation. In this case, the node SHOULD pass the information
downstream unchanged.

The recording of the Label subobject in the ROUTE_RECORD object is
controlled by the label-recording-desired flag in the
SESSION_ATTRIBUTE object. Since the Label subobject is not needed
for all applications, it is not automatically recorded. The flag
allows applications to request this only when needed.

The contents of the Session Name field are a string, typically of
displayable characters. The Length MUST always be a multiple of 4
and MUST be at least 8. For an object length that is not a multiple
of 4, the object is padded with trailing NULL characters. The Name
Length field contains the actual string length.

4.7.4. Resource Affinity Procedures

Resource classes and resource class affinities are described in [3].
In this document we use the briefer term resource affinities for the
latter term. Resource classes can be associated with links and
advertised in routing protocols. Resource class affinities are used
by RSVP in two ways. In order to be validated a link MUST pass the
three tests below. If the test fails a PathErr with the code

"policy
   control failure" SHOULD be sent.

   When a new reservation is considered for admission over a strict
node
   in an ERO, a node MAY validate the resource affinities with the
   resource classes of that link. When a node is choosing links in
   order to extend a loose node of an ERO, the node MUST validate the
   resource classes of those links against the resource affinities.
If
   no acceptable links can be found to extend the ERO, the node SHOULD
   send a PathErr message with an error code of "Routing Problem" and
an
   error value of "no route available toward destination".

   In order to be validated a link MUST pass the following three
tests.

   To precisely describe the tests use the definitions in the object
   description above. We also define

      Link-attr   A 32-bit vector representing attributes
associated
                     with a link.

   The three tests are

      1. Exclude-any

          This test excludes a link from consideration if the link
carries any
          of the attributes in the set.

          (link-attr & exclude-any) == 0

      2. Include-any

This test accepts a link if the link carries any of the attributes in the set.

(include-any == 0) | ((link-attr & include-any) != 0)

3. Include-all

This test accepts a link only if the link carries all of the attributes in the set.

(include-all == 0) | (((link-attr & include-all) ^ include-all) == 0)

For a link to be acceptable, all three tests MUST pass. If the test fails a error message

If a Path message contains multiple SESSION_ATTRIBUTE objects, only the first SESSION_ATTRIBUTE object is meaningful. Subsequent SESSION_ATTRIBUTE objects can be ignored and need not be forwarded.

All RSVP routers, whether they support the SESSION_ATTRIBUTE object or not, SHALL forward the object unmodified. The presence of non-RSVP routers anywhere between senders and receivers has no impact on this object.

4.8. Tspec and Flowspec Object for Class-of-Service Service

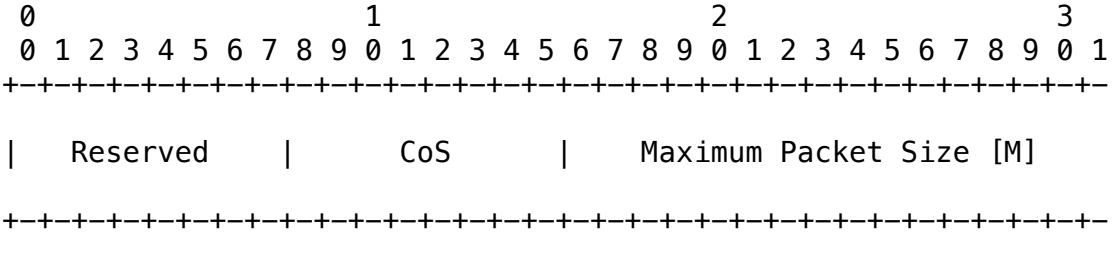
An LSP may not need bandwidth reservations or QoS guarantees. Such LSPs can be used to deliver best-effort traffic, even if RSVP is used for setting up LSPs. When resources do not have to be allocated to the LSP, the Class-of-Service service SHOULD be used.

The Class-of-Service FLOWSPEC allows indication of a Class of Service (CoS) value that should be used when handling data packets associated with the request.

The same format is used both for SENDER_TSPEC object and FLOWSPEC objects. The formats are:

Class-of-Service SENDER_TSPEC object: Class = 12, C-Type = 3

Class-of-Service FLOWSPEC object: Class = 9, C-Type = 3



Reserved

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

CoS

Indicates the Class of Service (CoS) of the data traffic associated with the request. A value of zero (0) indicates that associated traffic is "Best-Effort". Specifically no service assurances are being requested from the network. The intent is to enable networks to support the IP ToS Octet as defined in RFC1349 [7]. It is noted that there is ongoing work within the IETF to update the use of the IP ToS Octet. In particular, RFC2474 [8], obsoletes RFC1349. However at this time the new uses of this field (now termed the DS byte) are still being defined. Non-zero values have local significance. The translation from a specific value to an allocation is a

local administrative decision.

M
This parameter is set in Resv messages by the receiver based on information in arriving RSVP SENDER_TSPEC objects. For shared reservations, the smallest value received across all associated senders is used. When the object is contained in Path messages, this parameter is updated at each hop with the lesser of the received value and the MTU of the outgoing interface.

There is no Adspec associated with the Class-of-Service SENDER_TSPEC.
Either the Adspec is omitted or an int-serv adspec with only the Default General Characterization Parameters is used.

5. Hello Extension

The RSVP Hello extension enables RSVP nodes to detect when a neighboring node is not reachable. The mechanism provides node to node failure detection. When such a failure is detected it is handled much the same as a link layer communication failure. This mechanism is intended to be used when notification of link layer failures is not available and unnumbered links are not used, or when the failure detection mechanisms provided by the link layer are not sufficient for timely node failure detection.

It should be noted that node failure detection is not the same as a link failure detection mechanism, particularly in the case of multiple parallel unnumbered links.

The Hello extension is specifically designed so that one side can use the mechanism while the other side does not. Neighbor failure detection may be initiated at any time. This includes when neighbors first learn about each other, or just when neighbors are sharing Resv or Path state.

The Hello extension is composed of a Hello message, a HELLO REQUEST object and a HELLO ACK object. Hello processing between two neighbors supports independent selection of, typically configured, failure detection intervals. Each neighbor can autonomously issue HELLO REQUEST objects. Each request is answered by an acknowledgment. Hello Messages also contain enough information so that one neighbor can suppress issuing hello requests and still perform neighbor failure detection. A Hello message may be included as a sub-message within a bundle message.

Neighbor failure detection is accomplished by collecting and storing a neighbor's "instance" value. If a change in value is seen or if the neighbor is not properly reporting the locally advertised value, then the neighbor is presumed to have reset. When a neighbor's value is seen to change or when communication is lost with a neighbor, then the instance value advertised to that neighbor is also changed. The HELLO objects provide a mechanism for polling for and providing an instance value. A poll request also includes the sender's instance value. This allows the receiver of a poll to optionally treat the poll as an implicit poll response. This optional handling is an optimization that can reduce the total number of polls and responses processed by a pair of neighbors. In all cases, when both sides support the optimization the result will be only one set of polls and responses per failure detection interval. Depending on selected intervals, the same benefit can occur even when only one neighbor supports the optimization.

5.1. Hello Message Format

Hello Messages are always sent between two RSVP neighbors. The IP source address is the IP address of the sending node. The IP destination address is the IP address of the neighbor node.

The HELLO mechanism is intended for use between immediate neighbors.
When HELLO messages are being the exchanged between immediate neighbors, the IP TTL field of all outgoing HELLO messages SHOULD be set to 1.

The Hello message has a Msg Type of 20. The Hello message format is as follows:

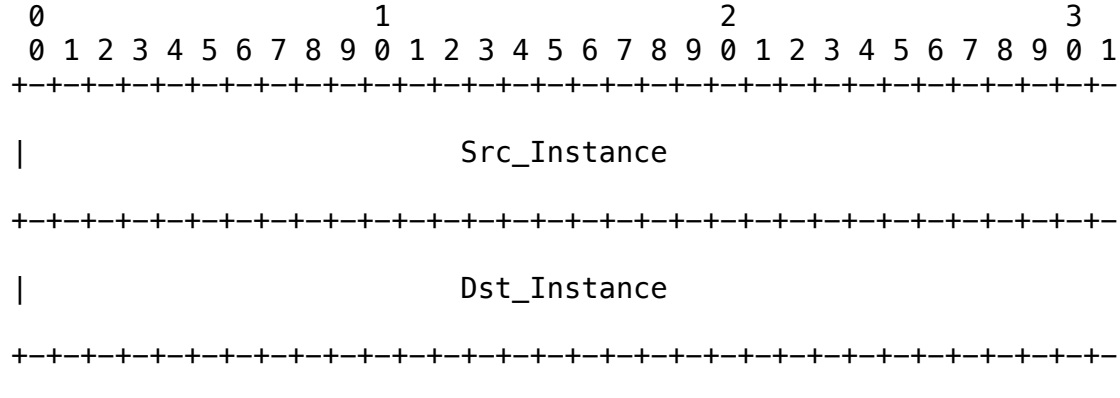
```
<Hello Message> ::= <Common Header> [ <INTEGRITY> ]
                          <HELLO>
```

5.2. HELLO Object formats

The HELLO Class is 22. There are two C_Types defined.

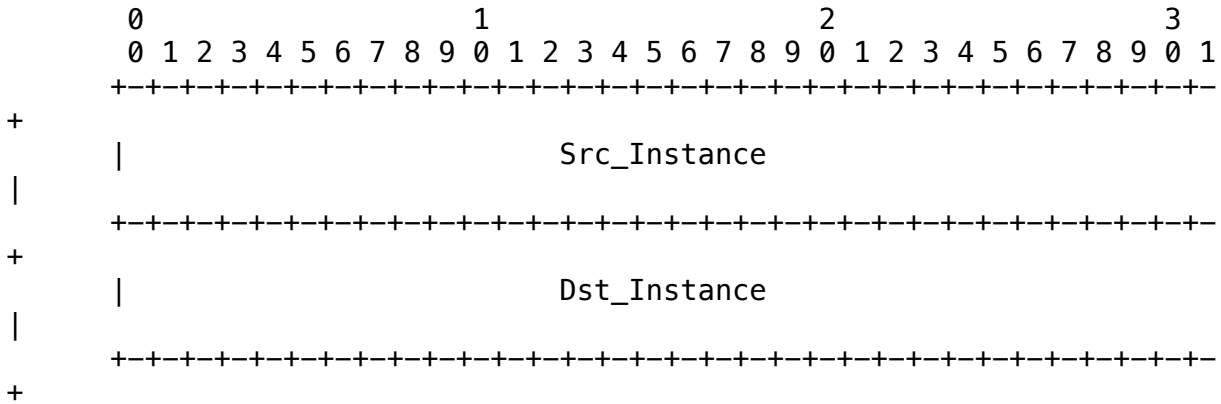
5.2.1. HELLO REQUEST object

Class = HELLO Class, C_Type = 1



5.2.2. HELLO ACK object

Class = HELLO Class, C_Type = 2



Src_Instance: 32 bits

a 32 bit value that represents the sender's instance. The advertiser maintains a per neighbor representation/value. This value MUST change when the sender is reset, when the node reboots, or when communication is lost to the neighboring node and otherwise remains the same. This field MUST NOT be set to

zero (0).

Dst_Instance: 32 bits

The most recently received Src_Instance value received from the neighbor. This field MUST be set to zero (0) when no value has ever been seen from the neighbor.

5.3. Hello Message Usage

The Hello Message is completely OPTIONAL. All messages may be ignored by nodes which do not wish to participate in Hello message processing. The balance of this section is written assuming that the receiver as well as the sender is participating. In particular, the use of MUST and SHOULD with respect to the receiver applies only to a node that supports Hello message processing.

A node periodically generates a Hello message containing a HELLO REQUEST object for each neighbor who's status is being tracked. The periodicity is governed by the hello_interval. This value MAY be configured on a per neighbor basis. The default value is 5 ms.

When generating a message containing a HELLO REQUEST object, the sender fills in the Src_Instance field with a value representing it's per neighbor instance. This value MUST NOT change while the agent is exchanging Hellos with the corresponding neighbor. The sender also fills in the Dst_Instance field with the Src_Instance value most

recently received from the neighbor. For reference, call this variable Neighbor_Src_Instance. If no value has ever been received from the neighbor or this node considers communication to the

neighbor to have been lost, the Neighbor_Src_Instance is set to zero (0). The generation of a message SHOULD be suppressed when a HELLO REQUEST object was received from the destination node within the prior hello_interval interval.

On receipt of a message containing a HELLO REQUEST object, the receiver MUST generate a Hello message containing a HELLO ACK object. The receiver SHOULD also verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the Neighbor_Src_Instance value is zero, and the Src_Instance field is non-zero, the Neighbor_Src_Instance is updated with the new value. If the value differs or the Src_Instance field is zero, then the node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO REQUEST object SHOULD also verify that the neighbor is reflecting back the receiver's Instance value. This is done by comparing the received Dst_Instance field with the Src_Instance field value most recently transmitted to that neighbor. If the neighbor continues to advertise a wrong non-zero value after a configured number of intervals, then the node MUST treat the neighbor as if communication has been lost.

On receipt of a message containing a HELLO ACK object, the receiver MUST verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the Neighbor_Src_Instance value is zero, and the Src_Instance field is non-zero, the Neighbor_Src_Instance is updated with the new value. If the value differs or the Src_Instance field is zero, then the node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO ACK object MUST also verify that the neighbor is reflecting back the receiver's Instance value. If the neighbor advertises a wrong value in the Dst_Instance field, then a node MUST treat the neighbor as if communication has been lost.

If no Instance values are received, via either REQUEST or ACK objects, from a neighbor within a configured number of

hello_intervals, then a node MUST presume that it cannot communicate with the neighbor. The default for this number is 3.5.

When communication is lost or presumed to be lost as described above, a node MAY re-initiate HELLOs. If a node does re-initiate it MUST use a Src_Instance value different than the one advertised in the

previous HELLO message. This new value MUST continue to be advertised to the corresponding neighbor until a reset or reboot occurs, or until another communication failure is detected. If a new instance value has not been received from the neighbor, then the node MUST advertise zero in the Dst_instance value field.

5.4. Multi-Link Considerations

As previously noted, the Hello extension is targeted at detecting node failures not per link failures. When there is only one link between neighboring nodes or when all links between a pair of nodes fail, the distinction between node and link failures is not really meaningful and handling of such failures has already been covered. When there are multiple links shared between neighbors, there are special considerations. When the links between neighbors are numbered, then Hellos MUST be run on each link and the previously described mechanisms apply.

When the links are unnumbered, link failure detection MUST be provided by some means other than Hellos. Each node SHOULD use a single Hello exchange with the neighbor. The case where all links have failed, is the same as the no received value case mentioned in the previous section.

5.5. Compatibility

The Hello extension does not affect the processing of any other

RSVP
message. The only effect is to allow a link (node) down event to
be
declared sooner than it would have been. RSVP response to that
condition is unchanged.

The Hello extension is fully backwards compatible. The Hello class
is assigned a class value of the form 0bbbbbbb. Depending on the
implementation, implementations that do not support the extension
will either silently discard Hello messages or will respond with an
"Unknown Object Class" error. In either case the sender will fail
to
see an acknowledgment for the issued Hello.

6. Security Considerations

In principle these extensions to RSVP pose no security exposures
over
and above RFC 2205[1]. However, there is a slight change in the
trust model. Traffic sent on a normal RSVP session can be filtered
according to source and destination addresses as well as port
numbers. In this specification, filtering occurs only on the basis
of an incoming label. For this reason an administration may wish
to
limit the domain over which LSP tunnels can be established. This
can
be accomplished by setting filters on various ports to deny action
on
a RSVP path message with a SESSION object of type LSP_TUNNEL_IPv4
(7)
or LSP_TUNNEL_IPv6 (8).

7. IANA Considerations

The responsible Internet authority (presently called the IANA) assigns values to RSVP protocol parameters. With the current document an EXPLICIT_ROUTE object and a ROUTE_RECORD object are defined. Each of these objects contain subobjects. This section defines the rules for the assignment of subobject numbers. This section uses the terminology of BCP 26 "Guidelines for Writing an IANA Considerations Section in RFCs".

EXPLICIT_ROUTE Subobject Type

EXPLICIT_ROUTE Subobject Type is a 7-bit number that identifies the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

ROUTE_RECORD Subobject Type

ROUTE_RECORD Subobject Type is an 8-bit number that identifies the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

8. Intellectual Property Considerations

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

9. Acknowledgments

This document contains ideas as well as text that have appeared in previous Internet Drafts. The authors of the current draft wish to thank the authors of those drafts. They are Steven Blake, Bruce Davie, Roch Guerin, Sanjay Kamat, Yakov Rekhter, Eric Rosen, and Arun Viswanathan. We also wish to thank Bora Akyol, Yoram Bernet and Alex Mondrus for their comments on this draft.

10. References

[1] Braden, R. et al., "Resource ReSerVation Protocol (RSVP) -- Version 1, Functional Specification", RFC 2205, September 1997.

[2] Rosen, E. et al., "Multiprotocol Label Switching Architeture", Internet Draft, draft-ietf-mpls-arch-06.txt, August 1999.

[3] Awduche, D. et al., "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[4] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.

[5] Rosen, E. et al., "MPLS Label Stack Encoding", Internet Draft, draft-ietf-mpls-label-encaps-07.txt, September 1999.

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

[7] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[8] Nichols, K. et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[9] Herzog, S., "Signaled Preemption Priority Policy Element", RFC 2751, January 2000.

[10] Awduche, D. et al., "Applicability Statement for Extensions to RSVP for LSP-Tunnels", draft-ietf-mpls-rsvp-tunnel-applicability-00.txt, September 1999.

[11] Wroclawski, J., "The Use of RSVP with IETF Integrated Services",

RFC 2210, September 1997.

[12] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[13] Mogul, J. & Deering, S., "Path MTU Discovery", RFC 1191, November 1990.

11. Authors' Addresses

Daniel O. Awduche
UUNET (MCI Worldcom), Inc
3060 Williams Drive
Fairfax, VA 22031
Voice: +1 703 208 5277
Email: awduche@uu.net

Lou Berger
LabN Consulting, LLC
Voice: +1 301 468 9228
Email: lberger@labn.net

Der-Hwa Gan
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
Voice: +1 650 526
Email: dhg@juniper.net

Tony Li
Procket Networks
3910 Freedom Circle, Ste. 102A
Santa Clara CA 95054
Email: tony1@home.net

Vijay Srinivasan
Cosine Communications, Inc.
1200 Bridge Parkway

Redwood City, CA 94065
Voice: +1 650 628 4892
Email: vsriniva@cosinecom.com

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

Network Working Group
Internet Draft
Expiration Date: August 2001

Daniel O. Awduche
Movaz Networks, Inc.

Lou Berger
Movaz Networks, Inc.

Der-Hwa Gan
Juniper Networks, Inc.

Tony Li
Procket Networks, Inc.

Vijay Srinivasan
Cosine Communications, Inc.

George Swallow
Cisco Systems, Inc.

February 2001

RSVP-TE: Extensions to RSVP for LSP Tunnels

draft-ietf-mpls-rsvp-lsp-tunnel-08.txt

Status of this Memo

Abstract

This document describes the use of RSVP, including all the necessary extensions, to establish label-switched paths (LSPs) in MPLS. Since the flow along an LSP is completely identified by the label applied

at the ingress node of the path, these paths may be treated as tunnels. A key application of LSP tunnels is traffic engineering with MPLS as specified in [3].

We propose several additional objects that extend RSVP, allowing the establishment of explicitly routed label switched paths using RSVP as a signaling protocol. The result is the instantiation of label-switched tunnels which can be automatically routed away from network failures, congestion, and bottlenecks.

Contents

1. Introduction

Section 2.9 of the MPLS architecture [2] defines a label distribution protocol as a set of procedures by which one Label Switched Router (LSR) informs another of the meaning of labels used to forward traffic between and through them. The MPLS architecture does not assume a single label distribution protocol. This document is a specification of extensions to RSVP for establishing label switched paths (LSPs) in Multi-protocol Label Switching (MPLS) networks.

Several of the new features described in this document were motivated by the requirements for traffic engineering over MPLS (see [3]). In particular, the extended RSVP protocol supports the instantiation of explicitly routed LSPs, with or without resource reservations. It

also supports smooth rerouting of LSPs, preemption, and loop detection.

The LSPs created with RSVP can be used to carry the "Traffic Trunks" described in [3]. The LSP which carries a traffic trunk and a traffic trunk are distinct though closely related concepts. For example, two LSPs between the same source and destination could be load shared to carry a single traffic trunk. Conversely several traffic trunks could be carried in the same LSP if, for instance, the LSP were capable of carrying several service classes. The applicability of these extensions is discussed further in [10].

Since the traffic that flows along a label-switched path is defined by the label applied at the ingress node of the LSP, these paths can be treated as tunnels, tunneling below normal IP routing and filtering mechanisms. When an LSP is used in this way we refer to it as an LSP tunnel.

LSP tunnels allow the implementation of a variety of policies related to network performance optimization. For example, LSP tunnels can be automatically or manually routed away from network failures, congestion, and bottlenecks. Furthermore, multiple parallel LSP tunnels can be established between two nodes, and traffic between the two nodes can be mapped onto the LSP tunnels according to local policy. Although traffic engineering (that is, performance optimization of operational networks) is expected to be an important application of this specification, the extended RSVP protocol can be used in a much wider context.

The purpose of this document is to describe the use of RSVP to establish LSP tunnels. The intent is to fully describe all the objects, packet formats, and procedures required to realize interoperable implementations. A few new objects are also defined that enhance management and diagnostics of LSP tunnels.

The document also describes a means of rapid node failure detection via a new HELLO message.

All objects and messages described in this specification are optional with respect to RSVP. This document discusses what happens when an object described here is not supported by a node.

Throughout this document, the discussion will be restricted to unicast label switched paths. Multicast LSPs are left for further study.

1.1. Background

Hosts and routers that support both RSVP [1] and Multi-Protocol Label Switching [2] can associate labels with RSVP flows. When MPLS and RSVP are combined, the definition of a flow can be made more flexible. Once a label switched path (LSP) is established, the traffic through the path is defined by the label applied at the ingress node of the LSP. The mapping of label to traffic can be accomplished using a number of different criteria. The set of packets that are assigned the same label value by a specific node are said to belong to the same forwarding equivalence class (FEC) (see [2]), and effectively define the "RSVP flow." When traffic is mapped onto a label-switched path in this way, we call the LSP an "LSP Tunnel". When labels are associated with traffic flows, it becomes possible for a router to identify the appropriate reservation state for a packet based on the packet's label value.

The signaling protocol model uses downstream-on-demand label distribution. A request to bind labels to a specific LSP tunnel is initiated by an ingress node through the RSVP Path message. For this purpose, the RSVP Path message is augmented with a LABEL_REQUEST object. Labels are allocated downstream and distributed (propagated upstream) by means of the RSVP Resv message. For this purpose, the RSVP Resv message is extended with a special LABEL object. The procedures for label allocation, distribution, binding, and stacking are described in subsequent sections of this document.

The signaling protocol model also supports explicit routing capability. This is accomplished by incorporating a simple EXPLICIT_ROUTE object into RSVP Path messages. The EXPLICIT_ROUTE object encapsulates a concatenation of hops which constitutes the explicitly routed path. Using this object, the paths taken by label-switched RSVP-MPLS flows can be pre-determined, independent of conventional IP routing. The explicitly routed path can be administratively specified, or automatically computed by a suitable entity based on QoS and policy requirements, taking into

consideration the prevailing network state. In general, path computation can be control-driven or data-driven. The mechanisms, processes, and algorithms used to compute explicitly routed paths are beyond the scope of this specification.

One useful application of explicit routing is traffic engineering. Using explicitly routed LSPs, a node at the ingress edge of an MPLS domain can control the path through which traffic traverses from itself, through the MPLS network, to an egress node. Explicit routing can be used to optimize the utilization of network resources and enhance traffic oriented performance characteristics.

The concept of explicitly routed label switched paths can be generalized through the notion of abstract nodes. An abstract node is a group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node. Using this concept of abstraction, an explicitly routed LSP can be specified as a sequence of IP prefixes or a sequence of Autonomous Systems.

The signaling protocol model supports the specification of an explicit path as a sequence of strict and loose routes. The combination of abstract nodes, and strict and loose routes significantly enhances the flexibility of path definitions.

An advantage of using RSVP to establish LSP tunnels is that it enables the allocation of resources along the path. For example, bandwidth can be allocated to an LSP tunnel using standard RSVP reservations and Integrated Services service classes [4].

While resource reservations are useful, they are not mandatory. Indeed, an LSP can be instantiated without any resource reservations whatsoever. Such LSPs without resource reservations can be used, for example, to carry best effort traffic. They can also be used in many other contexts, including implementation of fall-back and recovery policies under fault conditions, and so forth.

1.2. 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 [6].

The reader is assumed to be familiar with the terminology in [1], [2] and [3].

Abstract Node

A group of nodes whose internal topology is opaque to the ingress node of the LSP. An abstract node is said to be simple if it contains only one physical node.

Explicitly Routed LSP

An LSP whose path is established by a means other than normal IP routing.

Label Switched Path

The path created by the concatenation of one or more label switched hops, allowing a packet to be forwarded by swapping labels from an MPLS node to another MPLS node. For a more precise definition see [2].

LSP

A Label Switched Path

LSP Tunnel

An LSP which is used to tunnel below normal IP routing and/or filtering mechanisms.

Traffic Engineered Tunnel (TE Tunnel)

A set of one or more LSP Tunnels which carries a traffic trunk.

Traffic Trunk

A set of flows aggregated by their service class and then placed on an LSP or set of LSPs called a traffic engineered tunnel. For further discussion see [3].

2. Overview

2.1. LSP Tunnels and Traffic Engineered Tunnels

According to [1], "RSVP defines a 'session' to be a data flow with a particular destination and transport-layer protocol." However, when RSVP and MPLS are combined, a flow or session can be defined with greater flexibility and generality. The ingress node of an LSP can use a variety of means to determine which packets are assigned a particular label. Once a label is assigned to a set of packets, the label effectively defines the "flow" through the LSP. We refer to such an LSP as an "LSP tunnel" because the traffic through it is opaque to intermediate nodes along the label switched path.

New RSVP SESSION, SENDER_TEMPLATE, and FILTER_SPEC objects, called LSP_TUNNEL_IPv4 and LSP_TUNNEL_IPv6 have been defined to support the LSP tunnel feature. The semantics of these objects, from the perspective of a node along the label switched path, is that traffic belonging to the LSP tunnel is identified solely on the basis of packets arriving from the PHOP or "previous hop" (see [1]) with the particular label value(s) assigned by this node to upstream senders to the session. In fact, the IPv4(v6) that appears in the object name only denotes that the destination address is an IPv4(v6) address. When we refer to these objects generically, we use the qualifier LSP_TUNNEL.

In some applications it is useful to associate sets of LSP tunnels. This can be useful during reroute operations or to spread a traffic trunk over multiple paths. In the traffic engineering application such sets are called traffic engineered tunnels (TE tunnels). To enable the identification and association of such LSP tunnels, two identifiers are carried. A tunnel ID is part of the SESSION object. The SESSION object uniquely defines a traffic engineered tunnel. The SENDER_TEMPLATE and FILTER_SPEC objects carry an LSP ID. The

SENDER_TEMPLATE (or FILTER_SPEC) object together with the SESSION object uniquely identifies an LSP tunnel

2.2. Operation of LSP Tunnels

This section summarizes some of the features supported by RSVP as extended by this document related to the operation of LSP tunnels. These include: (1) the capability to establish LSP tunnels with or without QoS requirements, (2) the capability to dynamically reroute an established LSP tunnel, (3) the capability to observe the actual route traversed by an established LSP tunnel, (4) the capability to identify and diagnose LSP tunnels, (5) the capability to preempt an

established LSP tunnel under administrative policy control, and (6) the capability to perform downstream-on-demand label allocation, distribution, and binding. In the following paragraphs, these features are briefly described. More detailed descriptions can be found in subsequent sections of this document.

To create an LSP tunnel, the first MPLS node on the path -- that is, the sender node with respect to the path -- creates an RSVP Path message with a session type of LSP_TUNNEL_IPv4 or LSP_TUNNEL_IPv6 and inserts a LABEL_REQUEST object into the Path message. The LABEL_REQUEST object indicates that a label binding for this path is requested and also provides an indication of the network layer protocol that is to be carried over this path. The reason for this is that the network layer protocol sent down an LSP cannot be assumed to be IP and cannot be deduced from the L2 header, which simply identifies the higher layer protocol as MPLS.

If the sender node has knowledge of a route that has high likelihood of meeting the tunnel's QoS requirements, or that makes efficient use of network resources, or that satisfies some policy criteria, the node can decide to use the route for some or all of its sessions. To do this, the sender node adds an EXPLICIT_ROUTE object to the RSVP Path message. The EXPLICIT_ROUTE object specifies the route as a sequence of abstract nodes.

If, after a session has been successfully established, the sender node discovers a better route, the sender can dynamically reroute the session by simply changing the EXPLICIT_ROUTE object. If problems

are encountered with an EXPLICIT_ROUTE object, either because it causes a routing loop or because some intermediate routers do not support it, the sender node is notified.

By adding a RECORD_ROUTE object to the Path message, the sender node can receive information about the actual route that the LSP tunnel traverses. The sender node can also use this object to request notification from the network concerning changes to the routing path. The RECORD_ROUTE object is analogous to a path vector, and hence can be used for loop detection.

Finally, a SESSION_ATTRIBUTE object can be added to Path messages to aid in session identification and diagnostics. Additional control information, such as setup and hold priorities, resource affinities (see [3]), and local-protection, are also included in this object.

Routers along the path may use the setup and hold priorities along with SENDER_TSPEC and any POLICY_DATA objects contained in Path messages as input to policy control. For instance, in the traffic engineering application, it is very useful to use the Path message as

a means of verifying that bandwidth exists at a particular priority along an entire path before pre-empting any lower priority reservations. If a Path message is allowed to progress when there are insufficient resources, the there is a danger that lower priority reservations downstream of this point will unnecessarily be pre-empted in a futile attempt to service this request.

When the EXPLICIT_ROUTE object (ERO) is present, the Path message is forwarded towards its destination along a path specified by the ERO. Each node along the path records the ERO in its path state block. Nodes may also modify the ERO before forwarding the Path message. In this case the modified ERO SHOULD be stored in the path state block in addition to the received ERO.

The LABEL_REQUEST object requests intermediate routers and receiver nodes to provide a label binding for the session. If a node is incapable of providing a label binding, it sends a PathErr message with an "unknown object class" error. If the LABEL_REQUEST object is not supported end to end, the sender node will be notified by the first node which does not provide this support.

The destination node of a label-switched path responds to a LABEL_REQUEST by including a LABEL object in its response RSVP Resv message. The LABEL object is inserted in the filter spec list immediately following the filter spec to which it pertains.

The Resv message is sent back upstream towards the sender, following the path state created by the Path message, in reverse order. Note that if the path state was created by use of an ERO, then the Resv message will follow the reverse path of the ERO.

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, 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. The label sent upstream in the LABEL object is the label which this node will use to identify incoming traffic associated with this LSP tunnel. This label also serves as shorthand for the Filter Spec. The node can now update its "Incoming Label Map" (ILM), which is used to map incoming labeled packets to a "Next Hop Label Forwarding Entry" (NHLFE), see [2].

When the Resv message propagates upstream to the sender node, a label-switched path is effectively established.

2.3. Service Classes

This document does not restrict the type of Integrated Service requests for reservations. However, an implementation SHOULD support the Controlled-Load service [4] and the Class-of-Service service, see Section 4.8.

2.4. Reservation Styles

The receiver node can select from among a set of possible reservation styles for each session, and each RSVP session must have a particular style. Senders have no influence on the choice of reservation style. The receiver can choose different reservation styles for different

LSPs.

An RSVP session can result in one or more LSPs, depending on the reservation style chosen.

Some reservation styles, such as FF, dedicate a particular reservation to an individual sender node. Other reservation styles, such as WF and SE, can share a reservation among several sender nodes. The following sections discuss the different reservation styles and their advantages and disadvantages. A more detailed discussion of reservation styles can be found in [1].

2.4.1. Fixed Filter (FF) Style

The Fixed Filter (FF) reservation style creates a distinct reservation for traffic from each sender that is not shared by other senders. This style is common for applications in which traffic from each sender is likely to be concurrent and independent. The total amount of reserved bandwidth on a link for sessions using FF is the sum of the reservations for the individual senders.

Because each sender has its own reservation, a unique label is assigned to each sender. This can result in a point-to-point LSP between every sender/receiver pair.

2.4.2. Wildcard Filter (WF) Style

With the Wildcard Filter (WF) reservation style, a single shared reservation is used for all senders to a session. The total reservation on a link remains the same regardless of the number of senders.

A single multipoint-to-point label-switched-path is created for all senders to the session. On links that senders to the session share, a single label value is allocated to the session. If there is only one sender, the LSP looks like a normal point-to-point connection. When multiple senders are present, a multipoint-to-point LSP (a reversed tree) is created.

This style is useful for applications in which not all senders send traffic at the same time. A phone conference, for example, is an application where not all speakers talk at the same time. If, however, all senders send simultaneously, then there is no means of getting the proper reservations made. Either the reserved bandwidth on links close to the destination will be less than what is required or then the reserved bandwidth on links close to some senders will be greater than what is required. This restricts the applicability of WF for traffic engineering purposes.

Furthermore, because of the merging rules of WF, EXPLICIT_ROUTE objects cannot be used with WF reservations. As a result of this issue and the lack of applicability to traffic engineering, use of WF is not considered in this document.

2.4.3. Shared Explicit (SE) Style

The Shared Explicit (SE) style allows a receiver to explicitly specify the senders to be included in a reservation. There is a single reservation on a link for all the senders listed. Because each sender is explicitly listed in the Resv message, different labels may be assigned to different senders, thereby creating separate LSPs.

SE style reservations can be provided using multipoint-to-point label-switched-path or LSP per sender. Multipoint-to-point LSPs may be used when path messages do not carry the EXPLICIT_ROUTE object, or when Path messages have identical EXPLICIT_ROUTE objects. In either of these cases a common label may be assigned.

Path messages from different senders can each carry their own ERO, and the paths taken by the senders can converge and diverge at any point in the network topology. When Path messages have differing EXPLICIT_ROUTE objects, separate LSPs for each EXPLICIT_ROUTE object must be established.

2.5. Rerouting Traffic Engineered Tunnels

One of the requirements for Traffic Engineering is the capability to reroute an established TE tunnel under a number of conditions, based on administrative policy. For example, in some contexts, an administrative policy may dictate that a given TE tunnel is to be rerouted when a more "optimal" route becomes available. Another important context when TE tunnel reroute is usually required is upon failure of a resource along the TE tunnel's established path. Under some policies, it may also be necessary to return the TE tunnel to its original path when the failed resource becomes re-activated.

In general, it is highly desirable not to disrupt traffic, or adversely impact network operations while TE tunnel rerouting is in progress. This adaptive and smooth rerouting requirement necessitates establishing a new LSP tunnel and transferring traffic from the old LSP tunnel onto it before tearing down the old LSP tunnel. This concept is called "make-before-break." A problem can arise because the old and new LSP tunnels might compete with each other for resources on network segments which they have in common. Depending on availability of resources, this competition can cause Admission Control to prevent the new LSP tunnel from being established. An advantage of using RSVP to establish LSP tunnels is that it solves this problem very elegantly.

To support make-before-break in a smooth fashion, it is necessary that on links that are common to the old and new LSPs, resources used by the old LSP tunnel should not be released before traffic is transitioned to the new LSP tunnel, and reservations should not be counted twice because this might cause Admission Control to reject the new LSP tunnel.

A similar situation can arise when one wants to increase the bandwidth of a TE tunnel. The new reservation will be for the full amount needed, but the actual allocation needed is only the delta between the new and old bandwidth. If policy is being applied to PATH messages by intermediate nodes, then a PATH message requesting too much bandwidth will be rejected. In this situation simply increasing the bandwidth request without changing the SENDER_TEMPLATE, could result in a tunnel being torn down, depending upon local policy.

The combination of the LSP_TUNNEL SESSION object and the SE reservation style naturally accommodates smooth transitions in bandwidth and routing. The idea is that the old and new LSP tunnels share resources along links which they have in common. The LSP_TUNNEL SESSION object is used to narrow the scope of the RSVP session to the particular TE tunnel in question. To uniquely identify a TE tunnel,

we use the combination of the destination IP address (an address of the node which is the egress of the tunnel), a Tunnel ID, and the tunnel ingress node's IP address, which is placed in the Extended Tunnel ID field.

During the reroute or bandwidth-increase operation, the tunnel ingress needs to appear as two different senders to the RSVP session. This is achieved by the inclusion of the "LSP ID", which is carried in the SENDER_TEMPLATE and FILTER_SPEC objects. Since the semantics of these objects are changed, a new C-Types are assigned.

To effect a reroute, the ingress node picks a new LSP ID and forms a new SENDER_TEMPLATE. The ingress node then creates a new ERO to define the new path. Thereafter the node sends a new Path Message using the original SESSION object and the new SENDER_TEMPLATE and ERO. It continues to use the old LSP and refresh the old Path message. On links that are not held in common, the new Path message is treated as a conventional new LSP tunnel setup. On links held in common, the shared SESSION object and SE style allow the LSP to be established sharing resources with the old LSP. Once the ingress node receives a Resv message for the new LSP, it can transition traffic to it and tear down the old LSP.

To effect a bandwidth-increase, a new Path Message with a new LSP_ID can be used to attempt a larger bandwidth reservation while the current LSP_ID continues to be refreshed to ensure that the reservation is not lost if the larger reservation fails.

2.6. Path MTU

Standard RSVP [1] and Int-Serv [11] provide the RSVP sender with the minimum MTU available between the sender and the receiver. This path MTU identification capability is also provided for LSPs established via RSVP.

Path MTU information is carried, depending on which is present, in the Integrated Services or Class-of-Service objects. When using Integrated Services objects, path MTU is provided based on the procedures defined in [11]. Path MTU identification when using Class-of-Service objects is defined in Section 4.8.

With standard RSVP, the path MTU information is used by the sender to check which IP packets exceed the path MTU. For packets that exceed the path MTU, the sender either fragments the packets or, when the IP datagram has the "Don't Fragment" bit set, issues an ICMP destination unreachable message. This path MTU related handling is also required for LSPs established via RSVP.

The following algorithm applies to all unlabeled IP datagrams and to any labeled packets which the node knows to be IP datagrams, to which labels need to be added before forwarding. For labeled packets the bottom of stack is found, the IP header examined.

Using the terminology defined in [5], an LSR MUST execute the following algorithm:

1. Let N be the number of bytes in the label stack (i.e, 4 times the number of label stack entries) including labels to be added by this node.

2. Let M be the smaller of the "Maximum Initially Labeled IP Datagram Size" or of (Path MTU - N).

When the size of an IPv4 datagram (without labels) exceeds the value of M,

If the DF bit is not set in the IPv4 header, then

(a) the datagram MUST be broken into fragments, each of whose size is no greater than M, and

(b) each fragment MUST be labeled and then forwarded.

If the DF bit is set in the IPv4 header, then

(a) the datagram MUST NOT be forwarded

(b) Create an ICMP Destination Unreachable Message:
i. set its Code field [12] to "Fragmentation Required and DF Set",
ii. set its Next-Hop MTU field [13] to M

(c) If possible, transmit the ICMP Destination Unreachable

Message to the source of the of the discarded datagram.

When the size of an IPv6 datagram (without labels) exceeds the value of M,

(a) the datagram MUST NOT be forwarded

(b) Create an ICMP Packet too Big Message with the Next-Hop link MTU field [14] set to M

(c) If possible, transmit the ICMP Packet too Big Message to the source of the of the discarded datagram.

3. LSP Tunnel related Message Formats

Five new objects are defined in this section:

Object name	Applicable RSVP messages
LABEL_REQUEST	Path
LABEL	Resv
EXPLICIT_ROUTE	Path
RECORD_ROUTE	Path, Resv
SESSION_ATTRIBUTE	Path

New C-Types are also assigned for the SESSION, SENDER_TEMPLATE, FILTER_SPEC, FLOWSPEC objects.

Detailed descriptions of the new objects are given in later sections. All new objects are OPTIONAL with respect to RSVP. An implementation can choose to support a subset of objects. However, the LABEL_REQUEST and LABEL objects are mandatory with respect to this specification.

The LABEL and RECORD_ROUTE objects, are sender specific. In Resv messages they MUST appear after the associated FILTER_SPEC and prior to any subsequent FILTER_SPEC.

The relative placement of EXPLICIT_ROUTE, LABEL_REQUEST, and SESSION_ATTRIBUTE objects is simply a recommendation. The ordering of these objects is not important, so an implementation MUST be

prepared to accept objects in any order.

3.1. Path Message

The format of the Path message is as follows:

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

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


3.2. Resv Message

The format of the Resv message is as follows:

```
<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> [ <RECORD_ROUTE> ]
                         | <FF flow descriptor list>
                         <FF flow descriptor>

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

```
                              [ <RECORD_ROUTE> ]

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

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

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

      Note:  LABEL and RECORD_ROUTE (if present), are bound to the
             preceding FILTER_SPEC.  No more than one LABEL and/or
             RECORD_ROUTE may follow each FILTER_SPEC.
```

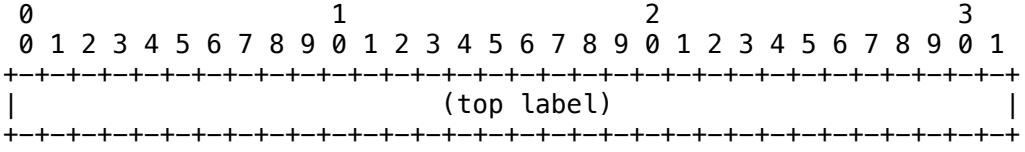
4. LSP Tunnel related Objects

4.1. Label Object

Labels MAY be carried in Resv messages. For the FF and SE styles, a label is associated with each sender. The label for a sender MUST immediately follow the FILTER_SPEC for that sender in the Resv message.

The LABEL object has the following format:

LABEL class = 16, C_Type = 1



The contents of a LABEL is a single label, encoded in 4 octets. Each generic MPLS label is an unsigned integer in the range 0 through 1048575. Generic MPLS labels and FR labels are encoded right aligned in 4 octets. ATM labels are encoded with the VPI right justified in bits 0-15 and the VCI right justified in bits 16-31.

4.1.1. Handling Label Objects in Resv messages

In MPLS a node may support multiple label spaces, perhaps associating a unique space with each incoming interface. For the purposes of the following discussion, the term "same label" means the identical label value drawn from the identical label space. Further, the following applies only to unicast sessions.

Labels received in Resv messages on different interfaces are always considered to be different even if the label value is the same.

4.1.1.1. Downstream

The downstream node selects a label to represent the flow. If a label range has been specified in the label request, the label MUST be drawn from that range. If no label is available the node sends a PathErr message with an error code of "routing problem" and an error value of "label allocation failure".

If a node receives a Resv message that has assigned the same label value to multiple senders, then that node MAY also assign a single value to those same senders or to any subset of those senders. Note that if a node intends to police individual senders to a session, it MUST assign unique labels to those senders.

In the case of ATM, one further condition applies. Some ATM nodes are not capable of merging streams. These nodes MAY indicate this by setting a bit in the label request to zero. The M-bit in the LABEL_REQUEST object of C-Type 2, label request with ATM label range, serves this purpose. The M-bit SHOULD be set by nodes which are merge capable. If for any senders the M-bit is not set, the downstream node MUST assign unique labels to those senders.

Once a label is allocated, the node formats a new LABEL object. The node then sends the new LABEL object as part of the Resv message to the previous hop. The LABEL object SHOULD be kept in the Reservation State Block. It is then used in the next Resv refresh event for formatting the Resv message.

A node is expected to send a Resv message before its refresh timers

expire if the contents of the LABEL object change.

4.1.1.2. Upstream

A node uses the label carried in the LABEL object as the outgoing label associated with the sender. The router allocates a new label and binds it to the incoming interface of this session/sender. This is the same interface that the router uses to forward Resv messages to the previous hops.

Several circumstance can lead to an unacceptable label.

1. the node is a merge incapable ATM switch but the downstream node has assigned the same label to two senders

2. The implicit null label was assigned, but the node is not capable of doing a penultimate pop for the associated L3PID

3. The assigned label is outside the requested label range

In any of these events the node send a ResvErr message with an error code of "routing problem" and an error value of "unacceptable label value".

4.1.2. Non-support of the Label Object

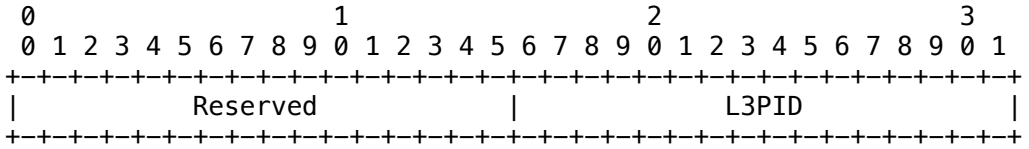
Under normal circumstances, a node should never receive a LABEL object in a Resv message unless it had included a LABEL_REQUEST object in the corresponding Path message. However, an RSVP router that does not recognize the LABEL object sends a ResvErr with the error code "Unknown object class" toward the receiver. This causes the reservation to fail.

4.2. Label Request Object

The Label Request Class is 19. Currently there three possible C_Types. Type 1 is a Label Request without label range. Type 2 is a label request with an ATM label range. Type 3 is a label request with a Frame Relay label range. The LABEL_REQUEST object formats are shown below.

4.2.1. Label Request without Label Range

Class = 19, C_Type = 1



Reserved

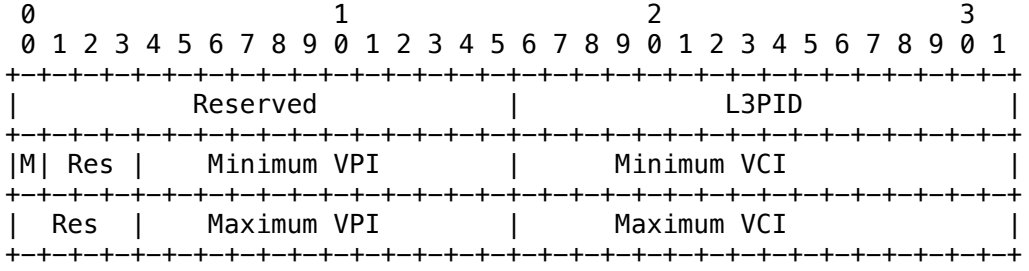
This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

4.2.2. Label Request with ATM Label Range

Class = 19, C_Type = 2



Reserved (Res)

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

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

M

Setting this bit to one indicates that the node is capable of merging in the data plane

Minimum VPI (12 bits)

This 12 bit field specifies the lower bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Minimum VCI (16 bits)

This 16 bit field specifies the lower bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VPI (12 bits)

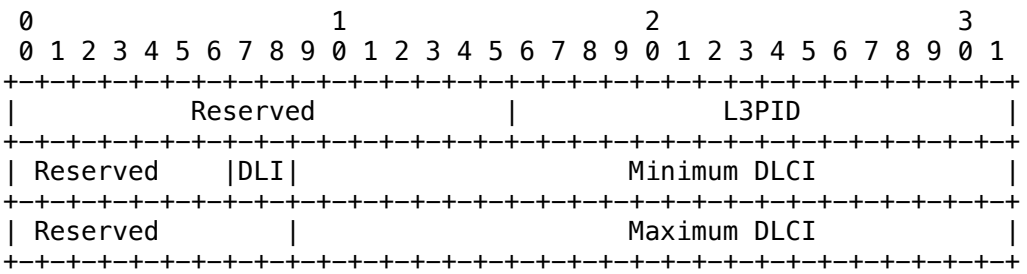
This 12 bit field specifies the upper bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

Maximum VCI (16 bits)

This 16 bit field specifies the upper bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16-bits it MUST be right justified in this field and preceding bits MUST be set to zero.

4.2.3. Label Request with Frame Relay Label Range

Class = 19, C_Type = 3



Reserved

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

L3PID

an identifier of the layer 3 protocol using this path. Standard Ethertype values are used.

DLI

DLCI Length Indicator. The number of bits in the DLCI. The following values are supported:

Len	DLCI bits
0	10
2	23

Minimum DLCI

This 23-bit field specifies the lower bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

Maximum DLCI

This 23-bit field specifies the upper bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI MUST be right justified in this field and unused bits MUST be set to 0.

4.2.4. Handling of LABEL_REQUEST

To establish an LSP tunnel the sender creates a Path message with a LABEL_REQUEST object. The LABEL_REQUEST object indicates that a label binding for this path is requested and provides an indication of the network layer protocol that is to be carried over this path. This permits non-IP network layer protocols to be sent down an LSP. This information can also be useful in actual label allocation, because some reserved labels are protocol specific, see [5].

The LABEL_REQUEST SHOULD be stored in the Path State Block, so that Path refresh messages will also contain the LABEL_REQUEST object. When the Path message reaches the receiver, the presence of the LABEL_REQUEST object triggers the receiver to allocate a label and to place the label in the LABEL object for the corresponding Resv message. If a label range was specified, the label MUST be allocated from that range. A receiver that accepts a LABEL_REQUEST object MUST include a LABEL object in Resv messages pertaining to that Path message. If a LABEL_REQUEST object was not present in the Path message, a node MUST NOT include a LABEL object in a Resv message for that Path message's session and PHOP.

A node that sends a LABEL_REQUEST object MUST be ready to accept and correctly process a LABEL object in the corresponding Resv messages.

A node that recognizes a LABEL_REQUEST object, but that is unable to support it (possibly because of a failure to allocate labels) SHOULD send a PathErr with the error code "Routing problem" and the error

value "MPLS label allocation failure." This includes the case where a label range has been specified and a label cannot be allocated from that range.

A node which receives and forwards a Path message each with a LABEL_REQUEST object, MUST copy the L3PID from the received LABEL_REQUEST object to the forwarded LABEL_REQUEST object.

If the receiver cannot support the protocol L3PID, it SHOULD send a PathErr with the error code "Routing problem" and the error value "Unsupported L3PID." This causes the RSVP session to fail.

4.2.5. Non-support of the Label Request Object

An RSVP router that does not recognize the LABEL_REQUEST object sends a PathErr with the error code "Unknown object class" toward the sender. An RSVP router that recognizes the LABEL_REQUEST object but does not recognize the C_Type sends a PathErr with the error code "Unknown object C_Type" toward the sender. This causes the path

setup to fail. The sender should notify management that a LSP cannot be established and possibly take action to continue the reservation without the LABEL_REQUEST.

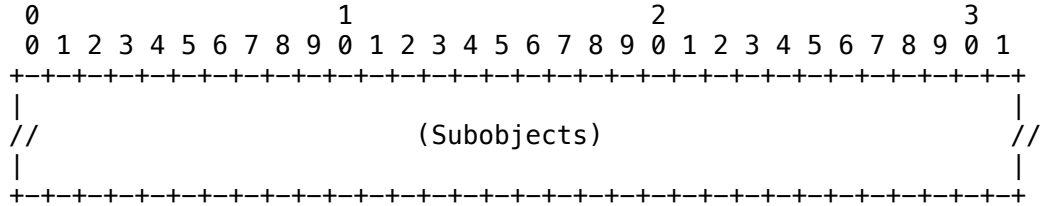
RSVP is designed to cope gracefully with non-RSVP routers anywhere between senders and receivers. However, obviously, non-RSVP routers cannot convey labels via RSVP. This means that if a router has a neighbor that is known to not be RSVP capable, the router MUST NOT advertise the LABEL_REQUEST object when sending messages that pass through the non-RSVP routers. The router SHOULD send a PathErr back to the sender, with the error code "Routing problem" and the error value "MPLS being negotiated, but a non-RSVP capable router stands in the path." This same message SHOULD be sent, if a router receives a LABEL_REQUEST object in a message from a non-RSVP capable router. See [1] for a description of how a downstream router can determine the presence of non-RSVP routers.

4.3. Explicit Route Object

Explicit routes are specified via the EXPLICIT_ROUTE object (ERO).

The Explicit Route Class is 20. Currently one C_Type is defined, Type 1 Explicit Route. The EXPLICIT_ROUTE object has the following format:

Class = 20, C_Type = 1



Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. The subobjects are defined in section 4.3.3 below.

If a Path message contains multiple EXPLICIT_ROUTE objects, only the first object is meaningful. Subsequent EXPLICIT_ROUTE objects MAY be ignored and SHOULD NOT be propagated.

4.3.1. Applicability

The EXPLICIT_ROUTE object is intended to be used only for unicast situations. Applications of explicit routing to multicast are a topic for further research.

The EXPLICIT_ROUTE object is to be used only when all routers along the explicit route support RSVP and the EXPLICIT_ROUTE object. The EXPLICIT_ROUTE object is assigned a class value of the form 0bbbbbbb. RSVP routers that do not support the object will therefore respond with an "Unknown Object Class" error.

4.3.2. Semantics of the Explicit Route Object

An explicit route is a particular path in the network topology. Typically, the explicit route is determined by a node, with the intent of directing traffic along that path.

An explicit route is described as a list of groups of nodes along the explicit route. In addition to the ability to identify specific nodes along the path, an explicit route can identify a group of nodes that must be traversed along the path. This capability allows the routing system a significant amount of local flexibility in fulfilling a request for an explicit route. This capability allows the generator of the explicit route to have imperfect information about the details of the path.

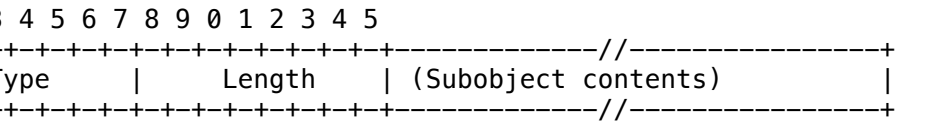
The explicit route is encoded as a series of subobjects contained in an EXPLICIT_ROUTE object. Each subobject identifies a group of nodes in the explicit route. An explicit route is thus a specification of groups of nodes to be traversed.

To formalize the discussion, we call each group of nodes an abstract node. Thus, we say that an explicit route is a specification of a set of abstract nodes to be traversed. If an abstract node consists of only one node, we refer to it as a simple abstract node.

As an example of the concept of abstract nodes, consider an explicit route that consists solely of Autonomous System number subobjects. Each subobject corresponds to an Autonomous System in the global topology. In this case, each Autonomous System is an abstract node, and the explicit route is a path that includes each of the specified Autonomous Systems. There may be multiple hops within each Autonomous System, but these are opaque to the source node for the explicit route.

4.3.3. Subobjects

The contents of an EXPLICIT_ROUTE object are a series of variable-length data items called subobjects. Each subobject has the form:



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

The Type indicates the type of contents of the subobject. Currently defined values are:

0	Reserved
1	IPv4 prefix
2	IPv6 prefix
32	Autonomous system number

Length

The Length contains the total length of the subobject in bytes, including the L, Type and Length fields. The Length MUST be at least 4, and MUST be a multiple of 4.

4.3.3.1. Strict and Loose Subobjects

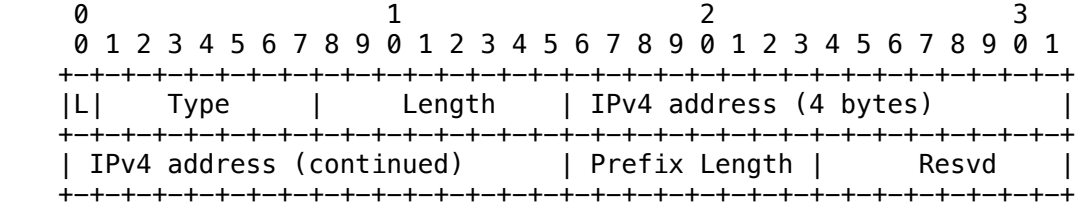
The L bit in the subobject is a one-bit attribute. If the L bit is set, then the value of the attribute is 'loose.' Otherwise, the value of the attribute is 'strict.' For brevity, we say that if the value of the subobject attribute is 'loose' then it is a 'loose subobject.' Otherwise, it's a 'strict subobject.' Further, we say that the abstract node of a strict or loose subobject is a strict or a loose node, respectively. Loose and strict nodes are always interpreted relative to their prior abstract nodes.

The path between a strict node and its preceding node MUST include

only network nodes from the strict node and its preceding abstract node.

The path between a loose node and its preceding node MAY include other network nodes that are not part of the strict node or its preceding abstract node.

4.3.3.2. Subobject 1: IPv4 prefix



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

An IPv4 address. This address is treated as a prefix based on the prefix length value below. Bits beyond the prefix are ignored on receipt and SHOULD be set to zero on transmission.

Prefix length

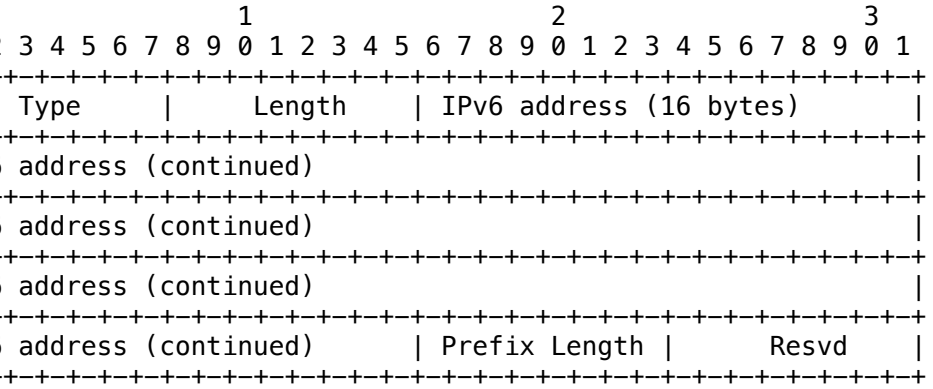
Length in bits of the IPv4 prefix

Padding

Zero on transmission. Ignored on receipt.

The contents of an IPv4 prefix subobject are a 4-octet IPv4 address, a 1-octet prefix length, and a 1-octet pad. The abstract node represented by this subobject is the set of nodes that have an IP address which lies within this prefix. Note that a prefix length of 32 indicates a single IPv4 node.

4.3.3.3. Subobject 2: IPv6 Prefix



L

The L bit is an attribute of the subobject. The L bit is set if the subobject represents a loose hop in the explicit route. If the bit is not set, the subobject represents a strict hop in the explicit route.

Type

0x02 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

An IPv6 address. This address is treated as a prefix based on the prefix length value below. Bits beyond the prefix are ignored on receipt and SHOULD be set to zero on transmission.

Prefix Length

Length in bits of the IPv6 prefix.

Padding

Zero on transmission. Ignored on receipt.

The contents of an IPv6 prefix subobject are a 16-octet IPv6 address, a 1-octet prefix length, and a 1-octet pad. The abstract node represented by this subobject is the set of nodes that have an IP address which lies within this prefix. Note that a prefix length of 128 indicates a single IPv6 node.

4.3.3.4. Subobject 32: Autonomous System Number

The contents of an Autonomous System (AS) number subobject are a 2-octet AS number. The abstract node represented by this subobject is the set of nodes belonging to the autonomous system.

The length of the AS number subobject is 4 octets.

4.3.4. Processing of the Explicit Route Object

4.3.4.1. Selection of the Next Hop

A node receiving a Path message containing an EXPLICIT_ROUTE object must determine the next hop for this path. This is necessary because the next abstract node along the explicit route might be an IP subnet or an Autonomous System. Therefore, selection of this next hop may involve a decision from a set of feasible alternatives. The criteria used to make a selection from feasible alternatives is implementation

dependent and can also be impacted by local policy, and is beyond the scope of this specification. However, it is assumed that each node will make a best effort attempt to determine a loop-free path. Note that paths so determined can be overridden by local policy.

To determine the next hop for the path, a node performs the following steps:

1) The node receiving the RSVP message MUST first evaluate the first subobject. If the node is not part of the abstract node described by the first subobject, it has received the message in error and SHOULD return a "Bad initial subobject" error. If there is no first subobject, the message is also in error and the system SHOULD return

a "Bad EXPLICIT_ROUTE object" error.

2) If there is no second subobject, this indicates the end of the explicit route. The EXPLICIT_ROUTE object SHOULD be removed from the Path message. This node may or may not be the end of the path. Processing continues with section 4.3.4.2, where a new EXPLICIT_ROUTE object MAY be added to the Path message.

3) Next, the node evaluates the second subobject. If the node is also a part of the abstract node described by the second subobject, then the node deletes the first subobject and continues processing with step 2, above. Note that this makes the second subobject into the first subobject of the next iteration and allows the node to identify the next abstract node on the path of the message after possible repeated application(s) of steps 2 and 3.

4) Abstract Node Border Case: The node determines whether it is topologically adjacent to the abstract node described by the second subobject. If so, the node selects a particular next hop which is a member of the abstract node. The node then deletes the first subobject and continues processing with section 4.3.4.2.

5) Interior of the Abstract Node Case: Otherwise, the node selects a next hop within the abstract node of the first subobject (which the node belongs to) that is along the path to the abstract node of the second subobject (which is the next abstract node). If no such path exists then there are two cases:

5a) If the second subobject is a strict subobject, there is an error and the node SHOULD return a "Bad strict node" error.

5b) Otherwise, if the second subobject is a loose subobject, the node selects any next hop that is along the path to the next abstract node. If no path exists, there is an error, and the node SHOULD return a "Bad loose node" error.

6) Finally, the node replaces the first subobject with any subobject that denotes an abstract node containing the next hop. This is necessary so that when the explicit route is received by the next hop, it will be accepted.

4.3.4.2. Adding subobjects to the Explicit Route Object

After selecting a next hop, the node MAY alter the explicit route in the following ways.

If, as part of executing the algorithm in section 4.3.4.1, the

EXPLICIT_ROUTE object is removed, the node MAY add a new EXPLICIT_ROUTE object.

Otherwise, if the node is a member of the abstract node for the first subobject, a series of subobjects MAY be inserted before the first subobject or MAY replace the first subobject. Each subobject in this series MUST denote an abstract node that is a subset of the current abstract node.

Alternately, if the first subobject is a loose subobject, an arbitrary series of subobjects MAY be inserted prior to the first subobject.

4.3.5. Loops

While the EXPLICIT_ROUTE object is of finite length, the existence of loose nodes implies that it is possible to construct forwarding loops during transients in the underlying routing protocol. This can be detected by the originator of the explicit route through the use of another opaque route object called the RECORD_ROUTE object. The

RECORD_ROUTE object is used to collect detailed path information and is useful for loop detection and for diagnostics.

4.3.6. Forward Compatibility

It is anticipated that new subobjects may be defined over time. A node which encounters an unrecognized subobject during its normal ERO processing sends a PathErr with the error code "Routing Error" and error value of "Bad Explicit Route Object" toward the sender. The EXPLICIT_ROUTE object is included, truncated (on the left) to the offending subobject. The presence of an unrecognized subobject which is not encountered in a node's ERO processing SHOULD be ignored. It is passed forward along with the rest of the remaining ERO stack.

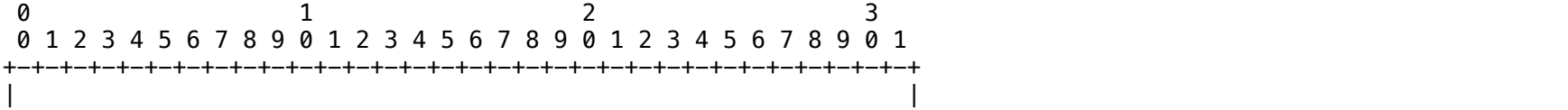
4.3.7. Non-support of the Explicit Route Object

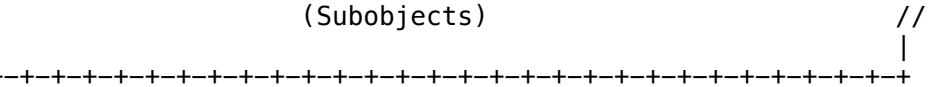
An RSVP router that does not recognize the EXPLICIT_ROUTE object sends a PathErr with the error code "Unknown object class" toward the sender. This causes the path setup to fail. The sender should notify management that a LSP cannot be established and possibly take action to continue the reservation without the EXPLICIT_ROUTE or via a different explicit route.

4.4. Record Route Object

Routes can be recorded via the RECORD_ROUTE object (RRO). Optionally, labels may also be recorded. The Record Route Class is 21. Currently one C_Type is defined, Type 1 Record Route. The RECORD_ROUTE object has the following format:

Class = 21, C_Type = 1





Subobjects

The contents of a RECORD_ROUTE object are a series of variable-length data items called subobjects. The subobjects are defined in section 4.4.1 below.

The RRO can be present in both RSVP Path and Resv messages. If a Path message contains multiple RROs, only the first RRO is meaningful. Subsequent RROs SHOULD be ignored and SHOULD NOT be propagated. Similarly, if in a Resv message multiple RROs are encountered following a FILTER_SPEC before another FILTER_SPEC is encountered, only the first RRO is meaningful. Subsequent RROs SHOULD be ignored and SHOULD NOT be propagated.

4.4.1. Subobjects

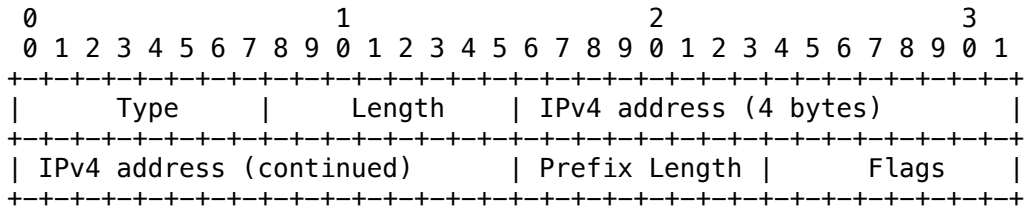
The contents of a RECORD_ROUTE object are a series of variable-length data items called subobjects. Each subobject has its own Length field. The length contains the total length of the subobject in bytes, including the Type and Length fields. The length MUST always be a multiple of 4, and at least 4.

Subobjects are organized as a last-in-first-out stack. The first subobject relative to the beginning of RRO is considered the top. The last subobject is considered the bottom. When a new subobject is added, it is always added to the top.

An empty RRO with no subobjects is considered illegal.

Three kinds of subobjects are currently defined.

4.4.1.1. Subobject 1: IPv4 address



Type

0x01 IPv4 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

A 32-bit unicast, host address. Any network-reachable interface address is allowed here. Illegal addresses, such as certain loopback addresses, SHOULD NOT be used.

Prefix length

32

Flags

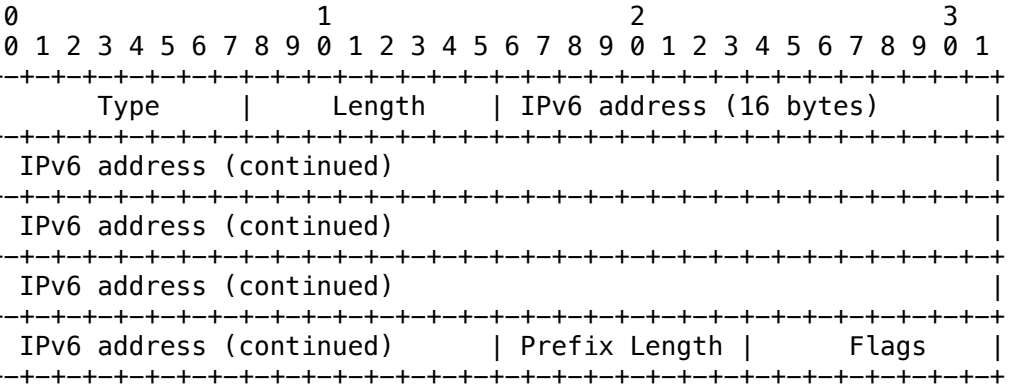
0x01 Local protection available

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).

4.4.1.2. Subobject 2: IPv6 address



Type

0x02 IPv6 address

Length

The Length contains the total length of the subobject in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

A 128-bit unicast host address.

Prefix length

128

Flags

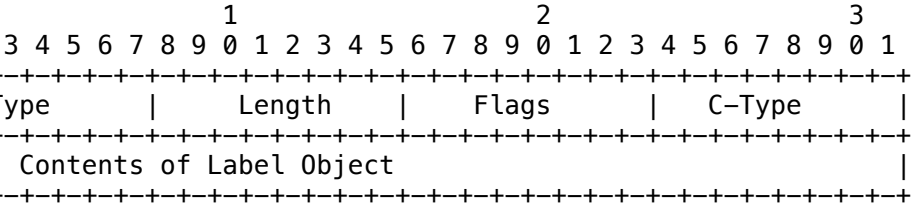
0x01 Local protection available

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 cooresponding Path nessage.

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).

4.4.1.3. Subobject 0x03, Label



Type

0x03 Label

Length

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

Flags

0x01 = Global label
This flag indicates that the label will be understood if received on any interface.

C-Type

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

Contents of Label Object

The contents of the Label Object. Copied from the Label Object

4.4.2. Applicability

Only the procedures for use in unicast sessions are defined here.

There are three possible uses of RRO in RSVP. First, an RRO can function as a loop detection mechanism to discover L3 routing loops, or loops inherent in the explicit route. The exact procedure for doing so is described later in this document.

Second, an RRO collects up-to-date detailed path information hop-by-hop about RSVP sessions, providing valuable information to the sender or receiver. Any path change (due to network topology changes) will be reported.

Third, RRO syntax is designed so that, with minor changes, the whole object can be used as input to the EXPLICIT_ROUTE object. This is useful if the sender receives RRO from the receiver in a Resv message, applies it to EXPLICIT_ROUTE object in the next Path message in order to "pin down session path".

4.4.3. Processing RRO

Typically, a node initiates an RSVP session by adding the RRO to the Path message. The initial RRO contains only one subobject - the sender's IP addresses. If the node also desires label recording, it sets the Label_Recording flag in the SESSION_ATTRIBUTE object.

When a Path message containing an RRO is received by an intermediate router, the router stores a copy of it in the Path State Block. The RRO is then used in the next Path refresh event for formatting Path messages. When a new Path message is to be sent, the router adds a new subobject to the RRO and appends the resulting RRO to the Path message before transmission.

The newly added subobject MUST be this router's IP address. The

address to be added SHOULD be the interface address of the outgoing Path messages. If there are multiple addresses to choose from, the decision is a local matter. However, it is RECOMMENDED that the same address be chosen consistently.

When the Label_Recording flag is set in the SESSION_ATTRIBUTE object, nodes doing route recording SHOULD include a Label Record subobject. If the node is using a global label space, then it SHOULD set the Global Label flag.

The Label Record subobject is pushed onto the RECORD_ROUTE object prior to pushing on the node's IP address. A node MUST NOT push on a Label Record subobject without also pushing on an IPv4 or IPv6 subobject.

Note that on receipt of the initial Path message, a node is unlikely to have a label to include. Once a label is obtained, the node SHOULD include the label in the RRO in the next Path refresh event.

If the newly added subobject causes the RRO to be too big to fit in a Path (or Resv) message, the RRO object SHALL be dropped from the message and message processing continues as normal. A PathErr (or ResvErr) message SHOULD be sent back to the sender (or receiver). An error code of "Notify" and an error value of "RRO too large for MTU" is used. If the receiver receives such a ResvErr, it SHOULD send a PathErr message with error code of "Notify" and an error value of "RRO notification".

A sender receiving either of these error values SHOULD remove the RRO from the Path message.

Nodes SHOULD resend the above PathErr or ResvErr message each n seconds where n is the greater of 15 and the refresh interval for the associated Path or RESV message. The node MAY apply limits and/or back-off timers to limit the number of messages sent.

An RSVP router can decide to send Path messages before its refresh time if the RRO in the next Path message is different from the previous one. This can happen if the contents of the RRO received from the previous hop router changes or if this RRO is newly added to (or deleted from) the Path message.

When the destination node of an RSVP session receives a Path message with an RRO, this indicates that the sender node needs route recording. The destination node initiates the RRO process by adding an RRO to Resv messages. The processing mirrors that of the Path messages. The only difference is that the RRO in a Resv message records the path information in the reverse direction.

Note that each node along the path will now have the complete route from source to destination. The Path RRO will have the route from the source to this node; the Resv RRO will have the route from this node to the destination. This is useful for network management.

A received Path message without an RRO indicates that the sender node no longer needs route recording. Subsequent Resv messages SHALL NOT contain an RRO.

4.4.4. Loop Detection

As part of processing an incoming RRO, an intermediate router looks into all subobjects contained within the RRO. If the router determines that it is already in the list, a forwarding loop exists.

An RSVP session is loop-free if downstream nodes receive Path messages or upstream nodes receive Resv messages with no routing loops detected in the contained RRO.

There are two broad classifications of forwarding loops. The first

class is the transient loop, which occurs as a normal part of operations as L3 routing tries to converge on a consistent forwarding path for all destinations. The second class of forwarding loop is the permanent loop, which normally results from network mis-configuration.

The action performed by a node on receipt of an RRO depends on the message type in which the RRO is received.

For Path messages containing a forwarding loop, the router builds and sends a "Routing problem" PathErr message, with the error value "loop detected," and drops the Path message. Until the loop is eliminated, this session is not suitable for forwarding data packets. How the loop eliminated is beyond the scope of this document.

For Resv messages containing a forwarding loop, the router simply drops the message. Resv messages should not loop if Path messages do not loop.

4.4.5. Forward Compatibility

New subobjects may be defined for the RRO. When processing an RRO, unrecognized subobjects SHOULD be ignored and passed on. When processing an RRO for loop detection, a node SHOULD parse over any unrecognized objects. Loop detection works by detecting subobjects which were inserted by the node itself on an earlier pass of the

object. This ensures that the subobjects necessary for loop detection are always understood.

4.4.6. Non-support of RRO

The RRO object is to be used only when all routers along the path support RSVP and the RRO object. The RRO object is assigned a class value of the form 0bbbbbbb. RSVP routers that do not support the object will therefore respond with an "Unknown Object Class" error.

4.5. Error Codes for ERO and RRO

In the processing described above, certain errors must be reported as either a "Routing Problem" or "Notify". The value of the "Routing Problem" error code is 24; the value of the "Notify" error code is 25.

The following defines error values for the Routing Problem Error Code:

Value	Error:
1	Bad EXPLICIT_ROUTE object
2	Bad strict node
3	Bad loose node
4	Bad initial subobject
5	No route available toward destination
6	Unacceptable label value
7	RRO indicated routing loops
8	MPLS being negotiated, but a non-RSVP-capable router stands in the path
9	MPLS label allocation failure
10	Unsupported L3PID

For the Notify Error Code, the 16 bits of the Error Value field are:

ss00 cccc cccc cccc

The high order bits are as defined under Error Code 1. (See [1]).

When ss = 00, the following subcodes are defined:

1 RRO too large for MTU

2 RRO notification

3 Tunnel locally repaired

4.6. Session, Sender Template, and Filter Spec Objects

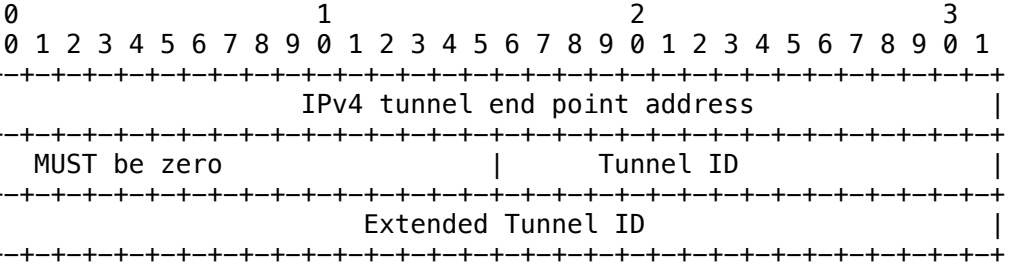
New C-Types are defined for the SESSION, SENDER_TEMPLATE and FILTER_SPEC objects.

The LSP_TUNNEL objects have the following format:

4.6.1. Session Object

4.6.1.1. LSP_TUNNEL_IPv4 Session Object

Class = SESSION, LSP_TUNNEL_IPv4 C-Type = 7



IPv4 tunnel end point address

IPv4 address of the egress node for the tunnel.

Tunnel ID

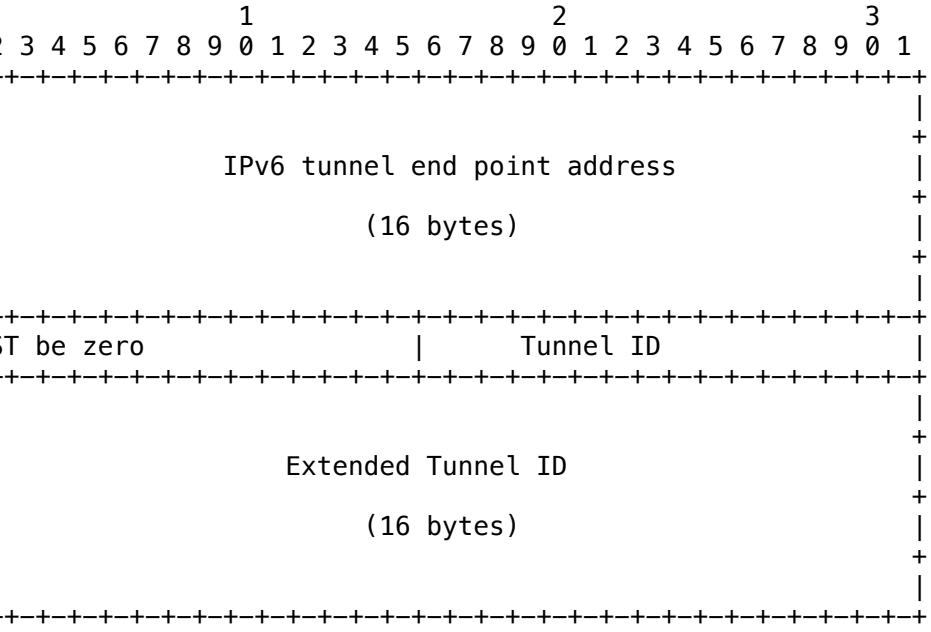
A 16-bit identifier used in the SESSION that remains constant over the life of the tunnel.

Extended Tunnel ID

A 32-bit identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv4 address here as a globally unique identifier.

4.6.1.2. LSP_TUNNEL_IPv6 Session Object

Class = SESSION, LSP_TUNNEL_IPv6 C_Type = 8



IPv6 tunnel end point address

IPv6 address of the egress node for the tunnel.

Tunnel ID

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

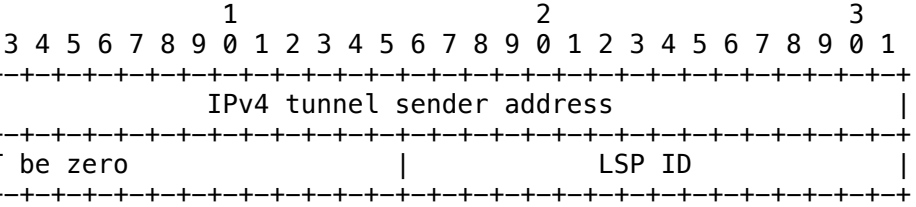
Extended Tunnel ID

A 16-byte identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingress-egress pair may place their IPv6 address here as a globally unique identifier.

4.6.2. Sender Template Object

4.6.2.1. LSP_TUNNEL_IPv4 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv4 C-Type = 7



IPv4 tunnel sender address

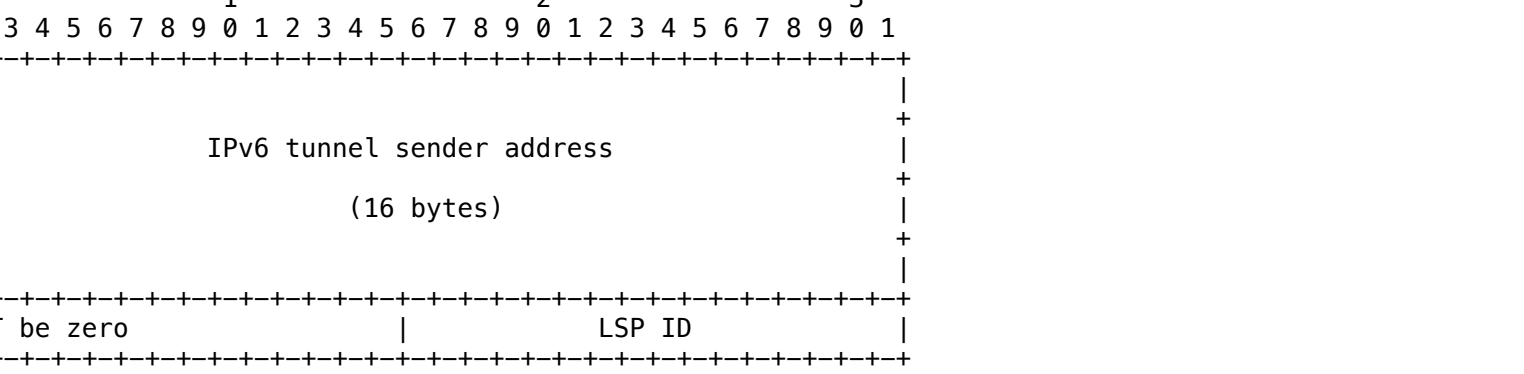
IPv4 address for a sender node

LSP ID

A 16-bit identifier used in the SENDER_TEMPLATE and the FILTER_SPEC that can be changed to allow a sender to share resources with itself.

4.6.2.2. LSP_TUNNEL_IPv6 Sender Template Object

Class = SENDER_TEMPLATE, LSP_TUNNEL_IPv6 C_Type = 8



IPv6 tunnel sender address

IPv6 address for a sender node

LSP ID

A 16-bit identifier used in the SENDER_TEMPLATE and the FILTER_SPEC that can be changed to allow a sender to share resources with itself.

4.6.3. Filter Specification Object

4.6.3.1. LSP_TUNNEL_IPv4 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv4 C-Type = 7

The format of the LSP_TUNNEL_IPv4 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv4 SENDER_TEMPLATE object.

4.6.3.2. LSP_TUNNEL_IPv6 Filter Specification Object

Class = FILTER SPECIFICATION, LSP_TUNNEL_IPv6 C_Type = 8

The format of the LSP_TUNNEL_IPv6 FILTER_SPEC object is identical to the LSP_TUNNEL_IPv6 SENDER_TEMPLATE object.

4.6.4. Reroute and Bandwidth Increase Procedure

This section describes how to setup a tunnel that is capable of maintaining resource reservations (without double counting) while it is being rerouted or while it is attempting to increase its bandwidth. In the initial Path message, the ingress node forms a SESSION object, assigns a Tunnel_ID, and places its IPv4 address in the Extended_Tunnel_ID It also forms a SENDER_TEMPLATE and assigns a LSP_ID. Tunnel setup then proceeds according to the normal procedure.

On receipt of the Path message, the egress node sends a Resv message with the STYLE Shared Explicit toward the ingress node.

When an ingress node with an established path wants to change that path, it forms a new Path message as follows. The existing SESSION object is used. In particular the Tunnel_ID and Extended_Tunnel_ID are unchanged. The ingress node picks a new LSP_ID to form a new SENDER_TEMPLATE. It creates an EXPLICIT_ROUTE object for the new route. The new Path message is sent. The ingress node refreshes both the old and new path messages

The egress node responds with a Resv message with an SE flow descriptor formatted as:

<FLOWSPEC><old_FILTER_SPEC><old_LABEL_OBJECT><new_FILTER_SPEC>
<new_LABEL_OBJECT>

(Note that if the PHOPs are different, then two messages are sent

each with the appropriate FILTER_SPEC and LABEL_OBJECT.)

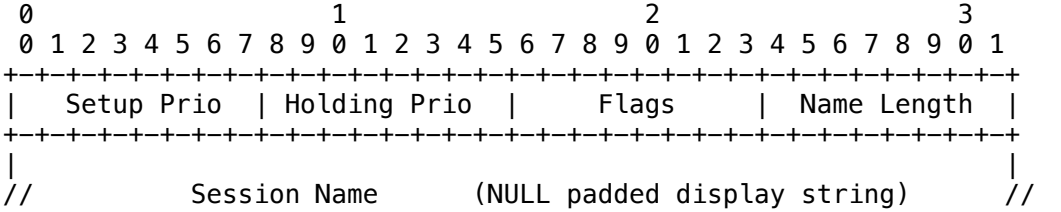
When the ingress node receives the Resv Message(s), it may begin using the new route. It SHOULD send a PathTear message for the old route.

4.7. Session Attribute Object

The Session Attribute Class is 207. Two C_Types are defined, LSP_TUNNEL, C-Type = 7 and LSP_TUNNEL_RA, C-Type = 1. The LSP_TUNNEL_RA C-Type includes all the same fields as the LSP_TUNNEL C-Type. Additionally it carries resource affinity information. The formats are as follows:

4.7.1. Format without resource affinities

SESSION_ATTRIBUTE class = 207, LSP_TUNNEL C-Type = 7



```
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Setup Priority

The priority of the session with respect to taking resources, in the range of 0 to 7. The value 0 is the highest priority. The Setup Priority is used in deciding whether this session can preempt another session.

Holding Priority

The priority of the session with respect to holding resources, in the range of 0 to 7. The value 0 is the highest priority. Holding Priority is used in deciding whether this session can be preempted by another session.

Flags

0x01 Local protection desired

This flag permits transit routers to use a local repair mechanism which may result in violation of the explicit route object. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

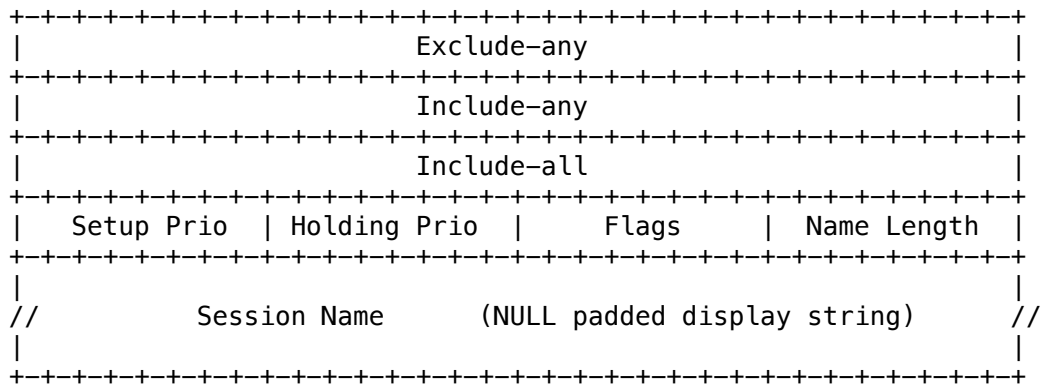
0x02 Label recording desired

This flag indicates that label information should be included when doing a route record.

```
          0x04  SE Style desired

                This flag indicates that the tunnel ingress node may
                choose to reroute this tunnel without tearing it down.
                A tunnel egress node SHOULD use the SE Style when
                responding with a Resv message.


      Name Length

         The length of the display string before padding, in bytes.

      Session Name

         A null padded string of characters.
```


4.7.2. Format with resource affinities

```
      SESSION_ATTRIBUTE class = 207, LSP_TUNNEL_RA C-Type = 1

      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
```



Exclude-any

A 32-bit vector representing a set of attribute filters associated with a tunnel any of which renders a link unacceptable.

Include-any

A 32-bit vector representing a set of attribute filters associated with a tunnel any of which renders a link acceptable (with respect to this test). A null set (all bits set to zero) automatically passes.

Include-all

A 32-bit vector representing a set of attribute filters associated with a tunnel all of which must be present for a link to be acceptable (with respect to this test). A null set (all bits set to zero) automatically passes.

Setup Priority

The priority of the session with respect to taking resources, in the range of 0 to 7. The value 0 is the highest priority. The Setup Priority is used in deciding whether this session can preempt another session.

Holding Priority

The priority of the session with respect to holding resources, in the range of 0 to 7. The value 0 is the highest priority. Holding Priority is used in deciding whether this session can be preempted by another session.

Flags

0x01 Local protection desired

This flag permits transit routers to use a local repair mechanism which may result in violation of the explicit route object. When a fault is detected on an adjacent downstream link or node, a transit router can reroute traffic for fast service restoration.

0x02 Label recording desired

This flag indicates that label information should be included when doing a route record.

0x04 SE Style desired

This flag indicates that the tunnel ingress node may choose to reroute this tunnel without tearing it down. A tunnel egress node SHOULD use the SE Style when responding with a Resv message.

Name Length

The length of the display string before padding, in bytes.

Session Name

A null padded string of characters.

4.7.3. Procedures applying to both C-Types

The support of setup and holding priorities is OPTIONAL. A node can recognize this information but be unable to perform the requested operation. The node SHOULD pass the information downstream unchanged.

As noted above, preemption is implemented by two priorities. The Setup Priority is the priority for taking resources. The Holding

Priority is the priority for holding a resource. Specifically, the Holding Priority is the priority at which resources assigned to this session will be reserved. The Setup Priority SHOULD never be higher than the Holding Priority for a given session.

The setup and holding priorities are directly analogous to the preemption and defending priorities as defined in [9]. While the interaction of these two objects is ultimately a matter of policy, the following default interaction is RECOMMENDED.

When both objects are present, the preemption priority policy element is used. A mapping between the priority spaces is defined as follows. A session attribute priority S is mapped to a preemption priority P by the formula P = 2^(14-2S). The reverse mapping is shown in the following table.

Preemption Priority	Session Attribute Priority
0 - 3	7
4 - 15	6
16 - 63	5
64 - 255	4
256 - 1023	3
1024 - 4095	2
4096 - 16383	1
16384 - 65535	0

When a new Path message is considered for admission, the bandwidth requested is compared with the bandwidth available at the priority specified in the Setup Priority.

If the requested bandwidth is not available a PathErr message is returned with an Error Code of 01, Admission Control Failure, and an Error Value of 0x0002. The first 0 in the Error Value indicates a globally defined subcode and is not informational. The 002 indicates "requested bandwidth unavailable".

If the requested bandwidth is less than the unused bandwidth then processing is complete. If the requested bandwidth is available, but is in use by lower priority sessions, then lower priority sessions (beginning with the lowest priority) MAY be pre-empted to free the necessary bandwidth.

When pre-emption is supported, each pre-empted reservation triggers a TC_Preempt() upcall to local clients, passing a subcode that indicates the reason. A ResvErr and/or PathErr with the code "Policy Control failure" SHOULD be sent toward the downstream receivers and upstream senders.

The support of local-protection is OPTIONAL. A node may recognize the local-protection Flag but may be unable to perform the requested operation. In this case, the node SHOULD pass the information downstream unchanged.

The recording of the Label subobject in the ROUTE_RECORD object is controlled by the label-recording-desired flag in the SESSION_ATTRIBUTE object. Since the Label subobject is not needed for all applications, it is not automatically recorded. The flag allows applications to request this only when needed.

The contents of the Session Name field are a string, typically of displayable characters. The Length MUST always be a multiple of 4 and MUST be at least 8. For an object length that is not a multiple of 4, the object is padded with trailing NULL characters. The Name Length field contains the actual string length.

4.7.4. Resource Affinity Procedures

Resource classes and resource class affinities are described in [3]. In this document we use the briefer term resource affinities for the latter term. Resource classes can be associated with links and advertised in routing protocols. Resource class affinities are used by RSVP in two ways. In order to be validated a link MUST pass the three tests below. If the test fails a PathErr with the code "policy control failure" SHOULD be sent.

When a new reservation is considered for admission over a strict node in an ERO, a node MAY validate the resource affinities with the resource classes of that link. When a node is choosing links in order to extend a loose node of an ERO, the node MUST validate the resource classes of those links against the resource affinities. If no acceptable links can be found to extend the ERO, the node SHOULD send a PathErr message with an error code of "Routing Problem" and an

error value of "no route available toward destination".

In order to be validated a link MUST pass the following three tests.

To precisely describe the tests use the definitions in the object description above. We also define

Link-attr A 32-bit vector representing attributes associated with a link.

The three tests are

1. Exclude-any

   This test excludes a link from consideration if the link carries any of the attributes in the set.

   (link-attr & exclude-any) == 0

2. Include-any

   This test accepts a link if the link carries any of the attributes in the set.

   (include-any == 0) | ((link-attr & include-any) != 0)

3. Include-all

   This test accepts a link only if the link carries all of the attributes in the set.

   (include-all == 0) | (((link-attr & include-all) ^ include-all) == 0)

For a link to be acceptable, all three tests MUST pass. If the test fails, the node SHOULD send a PathErr message with an error code of "Routing Problem" and an error value of "no route available toward destination".

If a Path message contains multiple SESSION_ATTRIBUTE objects, only the first SESSION_ATTRIBUTE object is meaningful. Subsequent SESSION_ATTRIBUTE objects can be ignored and need not be forwarded.

All RSVP routers, whether they support the SESSION_ATTRIBUTE object or not, SHALL forward the object unmodified. The presence of non-RSVP routers anywhere between senders and receivers has no impact on this object.

5. Hello Extension

The RSVP Hello extension enables RSVP nodes to detect when a neighboring node is not reachable. The mechanism provides node to node failure detection. When such a failure is detected it is handled much the same as a link layer communication failure. This mechanism is intended to be used when notification of link layer failures is not available and unnumbered links are not used, or when the failure detection mechanisms provided by the link layer are not sufficient for timely node failure detection.

It should be noted that node failure detection is not the same as a link failure detection mechanism, particularly in the case of multiple parallel unnumbered links.

The Hello extension is specifically designed so that one side can use the mechanism while the other side does not. Neighbor failure detection may be initiated at any time. This includes when neighbors first learn about each other, or just when neighbors are sharing Resv or Path state.

The Hello extension is composed of a Hello message, a HELLO REQUEST object and a HELLO ACK object. Hello processing between two neighbors supports independent selection of, typically configured, failure detection intervals. Each neighbor can autonomously issue HELLO REQUEST objects. Each request is answered by an acknowledgment. Hello Messages also contain enough information so that one neighbor can suppress issuing hello requests and still perform neighbor failure detection. A Hello message may be included as a sub-message within a bundle message.

Neighbor failure detection is accomplished by collecting and storing a neighbor's "instance" value. If a change in value is seen or if the neighbor is not properly reporting the locally advertised value, then the neighbor is presumed to have reset. When a neighbor's value is seen to change or when communication is lost with a neighbor, then the instance value advertised to that neighbor is also changed. The HELLO objects provide a mechanism for polling for and providing an instance value. A poll request also includes the sender's instance value. This allows the receiver of a poll to optionally treat the poll as an implicit poll response. This optional handling is an optimization that can reduce the total number of polls and responses processed by a pair of neighbors. In all cases, when both sides support the optimization the result will be only one set of polls and responses per failure detection interval. Depending on selected intervals, the same benefit can occur even when only one neighbor supports the optimization.

5.1. Hello Message Format

Hello Messages are always sent between two RSVP neighbors. The IP source address is the IP address of the sending node. The IP destination address is the IP address of the neighbor node.

The HELLO mechanism is intended for use between immediate neighbors. When HELLO messages are being the exchanged between immediate neighbors, the IP TTL field of all outgoing HELLO messages SHOULD be set to 1.

The Hello message has a Msg Type of 20. The Hello message format is as follows:

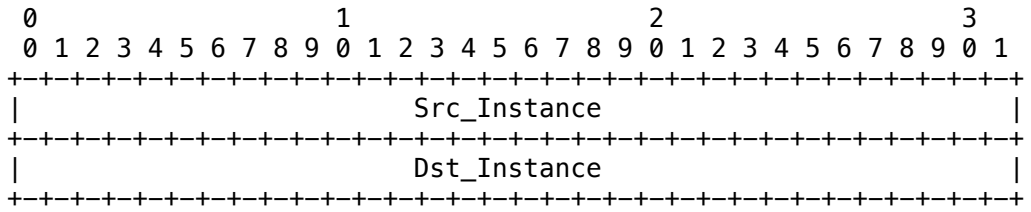
```
      <Hello Message> ::= <Common Header> [ <INTEGRITY> ]
                          <HELLO>
```

5.2. HELLO Object formats

The HELLO Class is 22. There are two C_Types defined.

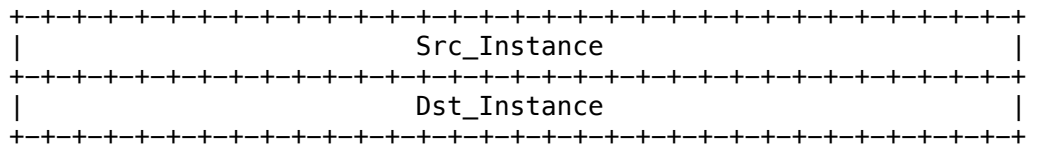
5.2.1. HELLO REQUEST object

Class = HELLO Class, C_Type = 1



5.2.2. HELLO ACK object

Class = HELLO Class, C_Type = 2



Src_Instance: 32 bits

a 32 bit value that represents the sender's instance. The advertiser maintains a per neighbor representation/value. This value MUST change when the sender is reset, when the node reboots, or when communication is lost to the neighboring node and otherwise remains the same. This field MUST NOT be set to zero (0).

Dst_Instance: 32 bits

The most recently received Src_Instance value received from the neighbor. This field MUST be set to zero (0) when no value has ever been seen from the neighbor.

5.3. Hello Message Usage

The Hello Message is completely OPTIONAL. All messages may be ignored by nodes which do not wish to participate in Hello message processing. The balance of this section is written assuming that the receiver as well as the sender is participating. In particular, the use of MUST and SHOULD with respect to the receiver applies only to a node that supports Hello message processing.

A node periodically generates a Hello message containing a HELLO REQUEST object for each neighbor who's status is being tracked. The periodicity is governed by the hello_interval. This value MAY be configured on a per neighbor basis. The default value is 5 ms.

When generating a message containing a HELLO REQUEST object, the sender fills in the Src_Instance field with a value representing it's per neighbor instance. This value MUST NOT change while the agent is exchanging Hellos with the corresponding neighbor. The sender also fills in the Dst_Instance field with the Src_Instance value most

recently received from the neighbor. For reference, call this variable Neighbor_Src_Instance. If no value has ever been received from the neighbor or this node considers communication to the neighbor to have been lost, the Neighbor_Src_Instance is set to zero (0). The generation of a message SHOULD be suppressed when a HELLO REQUEST object was received from the destination node within the prior hello_interval interval.

On receipt of a message containing a HELLO REQUEST object, the receiver MUST generate a Hello message containing a HELLO ACK object. The receiver SHOULD also verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the Neighbor_Src_Instance value is zero, and the Src_Instance field is non-zero, the Neighbor_Src_Instance is updated with the new value. If the value differs or the Src_Instance field is zero, then the node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO REQUEST object SHOULD also verify that the neighbor is reflecting back the receiver's Instance value. This is done by comparing the received Dst_Instance field with the Src_Instance field value most recently transmitted to that neighbor. If the neighbor continues to advertise a wrong non-zero value after a configured number of intervals, then the node MUST treat the neighbor as if communication has been lost.

On receipt of a message containing a HELLO ACK object, the receiver MUST verify that the neighbor has not reset. This is done by comparing the sender's Src_Instance field value with the previously received value. If the Neighbor_Src_Instance value is zero, and the Src_Instance field is non-zero, the Neighbor_Src_Instance is updated with the new value. If the value differs or the Src_Instance field is zero, then the node MUST treat the neighbor as if communication has been lost.

The receiver of a HELLO ACK object MUST also verify that the neighbor is reflecting back the receiver's Instance value. If the neighbor advertises a wrong value in the Dst_Instance field, then a node MUST treat the neighbor as if communication has been lost.

If no Instance values are received, via either REQUEST or ACK objects, from a neighbor within a configured number of hello_intervals, then a node MUST presume that it cannot communicate with the neighbor. The default for this number is 3.5.

When communication is lost or presumed to be lost as described above, a node MAY re-initiate HELLOs. If a node does re-initiate it MUST use a Src_Instance value different than the one advertised in the

previous HELLO message. This new value MUST continue to be advertised to the corresponding neighbor until a reset or reboot occurs, or until another communication failure is detected. If a new instance value has not been received from the neighbor, then the node MUST advertise zero in the Dst_instance value field.

5.4. Multi-Link Considerations

As previously noted, the Hello extension is targeted at detecting node failures not per link failures. When there is only one link between neighboring nodes or when all links between a pair of nodes fail, the distinction between node and link failures is not really meaningful and handling of such failures has already been covered. When there are multiple links shared between neighbors, there are special considerations. When the links between neighbors are numbered, then Hellos MUST be run on each link and the previously described mechanisms apply.

When the links are unnumbered, link failure detection MUST be provided by some means other than Hellos. Each node SHOULD use a single Hello exchange with the neighbor. The case where all links have failed, is the same as the no received value case mentioned in the previous section.

5.5. Compatibility

The Hello extension does not affect the processing of any other RSVP message. The only effect is to allow a link (node) down event to be declared sooner than it would have been. RSVP response to that condition is unchanged.

The Hello extension is fully backwards compatible. The Hello class is assigned a class value of the form 0bbbbbbb. Depending on the implementation, implementations that do not support the extension will either silently discard Hello messages or will respond with an "Unknown Object Class" error. In either case the sender will fail to see an acknowledgment for the issued Hello.

6. Security Considerations

In principle these extensions to RSVP pose no security exposures over and above RFC 2205[1]. However, there is a slight change in the trust model. Traffic sent on a normal RSVP session can be filtered according to source and destination addresses as well as port numbers. In this specification, filtering occurs only on the basis of an incoming label. For this reason an administration may wish to limit the domain over which LSP tunnels can be established. This can be accomplished by setting filters on various ports to deny action on a RSVP path message with a SESSION object of type LSP_TUNNEL_IPv4 (7) or LSP_TUNNEL_IPv6 (8).

7. IANA Considerations

IANA assigns values to RSVP protocol parameters. Within the current document an EXPLICIT_ROUTE object and a ROUTE_RECORD object are defined. Each of these objects contain subobjects. This section defines the rules for the assignment of subobject numbers. This section uses the terminology of BCP 26 "Guidelines for Writing an IANA Considerations Section in RFCs" [15].

EXPLICIT_ROUTE Subobject Type

EXPLICIT_ROUTE Subobject Type is a 7-bit number that identifies the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

Following the policies outlined in [15], subobject types in the range 0x00 - 0x3F are allocated through an IETF Consensus action, codes in the range 00x40 - 0x5F are allocated as First Come First Served, and codes in the range 0x60 - 0x7F are reserved for Private Use.

ROUTE_RECORD Subobject Type

ROUTE_RECORD Subobject Type is an 8-bit number that identifies the function of the subobject. There are no range restrictions. All possible values except zero are available for assignment.

Following the policies outlined in [15], subobject types in the range 0x00 - 0x7F are allocated through an IETF Consensus action, codes in the range 00x80 - 0xBF are allocated as First Come First Served, and codes in the range 0xC0 - 0xFF are reserved for Private Use.

8. Intellectual Property Considerations

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

9. Acknowledgments

This document contains ideas as well as text that have appeared in previous Internet Drafts. The authors of the current draft wish to thank the authors of those drafts. They are Steven Blake, Bruce Davie, Roch Guerin, Sanjay Kamat, Yakov Rekhter, Eric Rosen, and Arun Viswanathan. We also wish to thank Bora Akyol, Yoram Bernet and Alex Mondrus for their comments on this draft.

10. References

[1] Braden, R. et al., "Resource ReSerVation Protocol (RSVP) -- Version 1, Functional Specification", RFC 2205, September 1997.

[2] Rosen, E. et al., "Multiprotocol Label Switching Architeture", Internet Draft, draft-ietf-mpls-arch-06.txt, August 1999.

[3] Awduche, D. et al., "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[4] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.

[5] Rosen, E. et al., "MPLS Label Stack Encoding", Internet Draft, draft-ietf-mpls-label-encaps-07.txt, September 1999.

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

[7] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[8] Nichols, K. et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[9] Herzog, S., "Signaled Preemption Priority Policy Element",

RFC 2751, January 2000.

[10] Awduche, D. et al., "Applicability Statement for Extensions to RSVP for LSP-Tunnels", draft-ietf-mpls-rsvp-tunnel-applicability-00.txt, September 1999.

[11] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

[12] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[13] Mogul, J. & Deering, S., "Path MTU Discovery", RFC 1191, November 1990.

[14] Conta, A. & Deering, S., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)", RFC 2463, December 1998.

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

11. Authors' Addresses

Daniel O. Awduche
Movaz Networks, Inc.
7926 Jones Branch Drive, Suite 615
McLean, VA 22102
Voice: +1 703 847 7350
Email: awduche@movaz.com

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

Der-Hwa Gan
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
Email: dhg@juniper.net

Tony Li
Procket Networks

3910 Freedom Circle, Ste. 102A
Santa Clara CA 95054
Email: tony1@home.net

Vijay Srinivasan
Cosine Communications, Inc.
1200 Bridge Parkway
Redwood City, CA 94065
Voice: +1 650 628 4892
Email: vsriniva@cosinecom.com

George Swallow
Cisco Systems, Inc.
250 Apollo Drive

Chelmsford, MA 01824
Voice: +1 978 244 8143
Email: swallow@cisco.com

Network Working Group D. Katz
Internet Draft Juniper Networks
Category: Standards Track D. Yeung
Expires: April 2003 Procket Networks
draft-katz-yeung-ospf-traffic-09.txt K. Kompella
Juniper Networks

October 2002

Traffic Engineering Extensions to OSPF Version 2
*** Draft ***

Status

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document describes extensions to the OSPF protocol version 2 to support intra-area Traffic Engineering, using Opaque Link State Advertisements.

Changes

(This section to be removed before publication).

Per comments from the OSPF WG mailing list, the following changes were made:

- State that operation over multi-access networks with more than two TE devices is not expressly forbidden.
- Fix figure in 2.3.1.
- Specify that a Remote Interface IP Address sub-TLV is optional for a multi-access link.

1. Introduction

This document specifies a method of adding traffic engineering capabilities to OSPF Version 2 [1]. The architecture of traffic engineering is described in [2]. The semantic content of the extensions is essentially identical to the corresponding extensions to IS-IS [3]. It is expected that the traffic engineering extensions to OSPF will continue to mirror those in IS-IS.

The extensions provide a way of describing the traffic engineering topology (including bandwidth and administrative constraints) and distributing this information within a given OSPF area. This topology does not necessarily match the regular routed topology,

though this proposal depends on Network LSAs to describe multiaccess links.

1.1. Applicability

Many of the extensions specified in this document are in response to the requirements stated in [2], and thus are referred to as "traffic engineering extensions", and are also commonly associated with MPLS Traffic Engineering. A more accurate (albeit bland) designation is "extended link attributes", as what is proposed is simply to add more attributes to links in OSPF advertisements.

The information made available by these extensions can be used to build an extended link state database just as router LSAs are used to build a "regular" link state database; the difference is that the extended link state database (referred to below as the traffic engineering database) has additional link attributes. Uses of the traffic engineering database include:

- monitoring the extended link attributes;
- local constraint-based source routing; and
- global traffic engineering.

For example, an OSPF-speaking device can participate in an OSPF area, build a traffic engineering database, and thereby report on the reservation state of links in that area.

In "local constraint-based source routing", a router R can compute a path from a source node A to a destination node B; typically, A is R itself, and B is specified by a "router address" (see below). This path may be subject to various constraints on the attributes of the links and nodes that the path traverses, e.g., use green links that have unreserved bandwidth of at least 10Mbps. This path could then be used to carry some subset of the traffic from A to B, forming a simple but effective means of traffic engineering. How the subset of

traffic is determined, and how the path is instantiated is beyond the scope of this document; suffice it to say that one means of defining the subset of traffic is "those packets whose IP destinations were learned from B", and one means of instantiating paths is using MPLS tunnels. As an aside, note that constraint-based routing can be NP-hard, or even unsolvable, depending on the nature of the attributes and constraints and thus many implementations will use heuristics. Consequently, we don't attempt to sketch an algorithm here.

Finally, for "global traffic engineering", a device can build a traffic engineering database, input a traffic matrix and an optimization function, crunch on the information, and thus compute optimal or near-optimal routing for the entire network. The device can subsequently monitor the traffic engineering topology and react to changes by recomputing the optimal routes.

1.2. Limitations

As mentioned above, this document specifies extensions and procedures for intra-area distribution of Traffic Engineering information. Methods for inter-area and inter-AS (Autonomous System) are not discussed here.

The extensions specified in this document capture the reservation state of point-to-point links. The reservation state of multiaccess links may not be accurately reflected, except in the special case that there are only two devices in the multiaccess subnetwork. Operation over multiaccess networks with more than two devices is not specifically prohibited. More accurate description of the reservation state of multi-access networks is for further study.

This document also does not support unnumbered links. This deficiency is addressed in [4]; see also [5] and [6].

1.3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in RFC 2119 [7].

2. LSA Format

2.1. LSA type

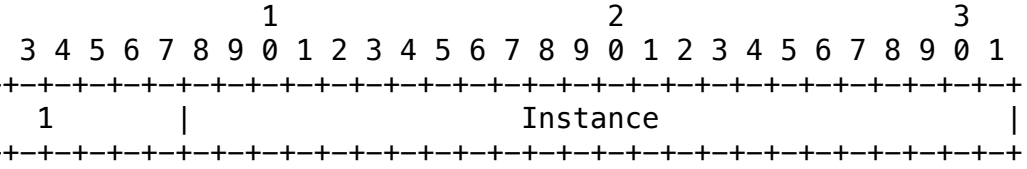
This extension makes use of the Opaque LSA [8].

Three types of Opaque LSAs exist, each of which has different flooding scope. This proposal uses only Type 10 LSAs, which have area flooding scope.

One new LSA is defined, the Traffic Engineering LSA. This LSA describes routers, point-to-point links, and connections to multiaccess networks (similar to a Router LSA). For traffic engineering purposes, the existing Network LSA suffices for describing multiaccess links, so no additional LSA is defined for this purpose.

2.2. LSA ID

The LSA ID of an Opaque LSA is defined as having eight bits of type and 24 bits of type-specific data. The Traffic Engineering LSA uses
type 1. The remaining 24 bits are the Instance field, as follows:

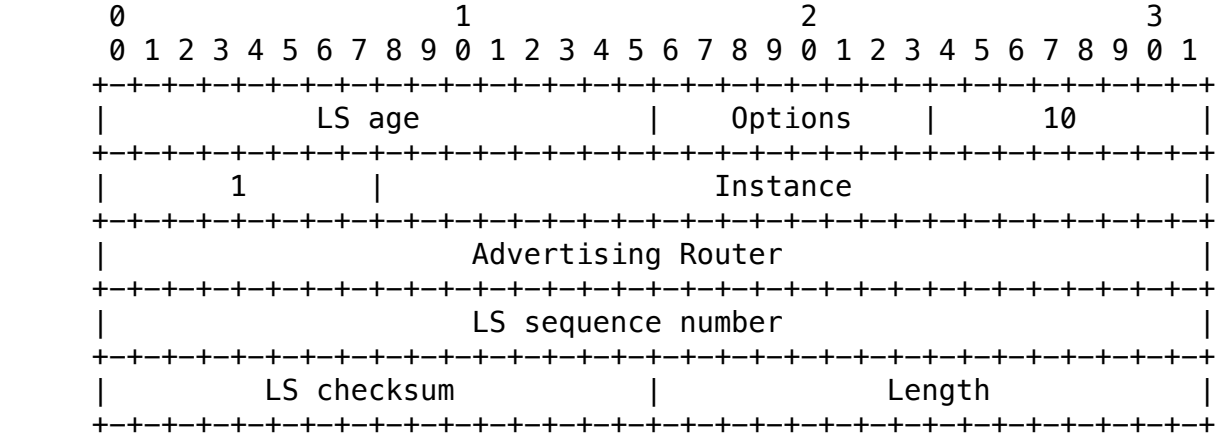


The Instance field is an arbitrary value used to maintain multiple Traffic Engineering LSAs. A maximum of 16777216 Traffic Engineering LSAs may be sourced by a single system. The LSA ID has no topological significance.

2.3. LSA Format Overview

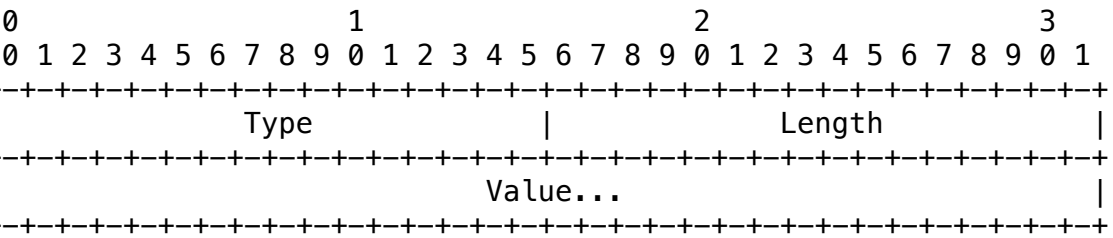
2.3.1. LSA Header

The Traffic Engineering LSA starts with the standard LSA header:



2.3.2. TLV Header

The LSA payload consists of one or more nested Type/Length/Value (TLV) triplets for extensibility. The format of each TLV is:



The Length field defines the length of the value portion in octets (thus a TLV with no value portion would have a length of zero). The TLV is padded to four-octet alignment; padding is not included in the length field (so a three octet value would have a length of three, but the total size of the TLV would be eight octets). Nested TLVs are also 32-bit aligned. Unrecognized types are ignored.

This memo defines Types 1 and 2. See the IANA Considerations section for allocation of new Types.

2.4. LSA payload details

An LSA contains one top-level TLV.

There are two top-level TLVs defined:

1 - Router Address
2 - Link

2.4.1. Router Address TLV

The Router Address TLV specifies a stable IP address of the advertising router that is always reachable if there is any connectivity to it. This is typically implemented as a "loopback address"; the key attribute is that the address does not become

unusable if an interface is down. In other protocols this is known as the "router ID," but for obvious reasons this nomenclature is avoided here. If a router advertises BGP routes with the BGP next hop attribute set to the BGP router ID, then the Router Address SHOULD be the same as the BGP router ID.

If IS-IS is also active in the domain, this address can also be used to compute the mapping between the OSPF and IS-IS topologies. For example, suppose a router R is advertising both IS-IS and OSPF Traffic Engineering LSAs, and suppose further that some router S is building a single Traffic Engineering Database (TED) based on both IS-IS and OSPF TE information. R may then appear as two separate nodes in S's TED; however, if both the IS-IS and OSPF LSAs generated by R contain the same Router Address, then S can determine that the IS-IS TE LSA and the OSPF TE LSA from R are indeed from a single router.

The router address TLV is type 1, and has a length of 4, and the value is the four octet IP address. It must appear in exactly one Traffic Engineering LSA originated by a router.

2.4.2. Link TLV

The Link TLV describes a single link. It is constructed of a set of sub-TLVs. There are no ordering requirements for the sub-TLVs.

Only one Link TLV shall be carried in each LSA, allowing for fine granularity changes in topology.

The Link TLV is type 2, and the length is variable.

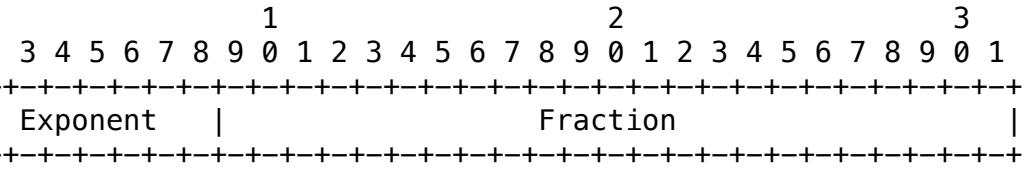
The following sub-TLVs are defined:

1 - Link type (1 octet)
2 - Link ID (4 octets)
3 - Local interface IP address (4 octets)
4 - Remote interface IP address (4 octets)
5 - Traffic engineering metric (4 octets)
6 - Maximum bandwidth (4 octets)
7 - Maximum reservable bandwidth (4 octets)
8 - Unreserved bandwidth (32 octets)
9 - Administrative group (4 octets)

This memo defines sub-Types 1 through 9. See the IANA Considerations section for allocation of new sub-Types.

The Link Type and Link ID sub-TLVs are mandatory, i.e., must appear exactly once. All other sub-TLVs defined here may occur at most once. These restrictions need not apply to future sub-TLVs. Unrecognized sub-TLVs are ignored.

Various values below use the (32 bit) IEEE Floating Point format. For quick reference, this format is as follows:



where S is the sign; Exponent is the exponent base 2 in "excess 127" notation; and Fraction is the mantissa - 1, with an implied binary point in front of it. Thus the above represents the value

(-1)**(S) * 2**(Exponent-127) * (1 + Fraction)

For more details, refer to [9].

2.5. Sub-TLV Details

2.5.1. Link Type

The Link Type sub-TLV defines the type of the link:

1 - Point-to-point
2 - Multiaccess

The Link Type sub-TLV is TLV type 1, and is one octet in length.

2.5.2. Link ID

The Link ID sub-TLV identifies the other end of the link. For point-to-point links, this is the Router ID of the neighbor. For multiaccess links, this is the interface address of the designated router. The Link ID is identical to the contents of the Link ID field in the Router LSA for these link types.

The Link ID sub-TLV is TLV type 2, and is four octets in length.

2.5.3. Local Interface IP Address

The Local Interface IP Address sub-TLV specifies the IP address(es) of the interface corresponding to this link. If there are multiple local addresses on the link, they are all listed in this sub-TLV.

The Local Interface IP Address sub-TLV is TLV type 3, and is 4N octets in length, where N is the number of local addresses.

2.5.4. Remote Interface IP Address

The Remote Interface IP Address sub-TLV specifies the IP address(es) of the neighbor's interface corresponding to this link. This and the local address are used to discern multiple parallel links between systems. If the Link Type of the link is Multiaccess, the Remote Interface IP Addess is set to 0.0.0.0; alternatively, an implementation MAY choose not to send this sub-TLV.

The Remote Interface IP Address sub-TLV is TLV type 4, and is 4N octets in length, where N is the number of neighbor addresses.

2.5.5. Traffic Engineering Metric

The Traffic Engineering Metric sub-TLV specifies the link metric for traffic engineering purposes. This metric may be different than the standard OSPF link metric. Typically, this metric is assigned by a network admistrator.

The Traffic Engineering Metric sub-TLV is TLV type 5, and is four octets in length.

2.5.6. Maximum Bandwidth

The Maximum Bandwidth sub-TLV specifies the maximum bandwidth that can be used on this link in this direction (from the system originating the LSA to its neighbor), in IEEE floating point format. This is the true link capacity. The units are bytes per second.

The Maximum Bandwidth sub-TLV is TLV type 6, and is four octets in length.

2.5.7. Maximum Reservable Bandwidth

The Maximum Reservable Bandwidth sub-TLV specifies the maximum bandwidth that may be reserved on this link in this direction, in IEEE floating point format. Note that this may be greater than the maximum bandwidth (in which case the link may be oversubscribed). This SHOULD be user-configurable; the default value should be the Maximum Bandwidth. The units are bytes per second.

The Maximum Reservable Bandwidth sub-TLV is TLV type 7, and is four octets in length.

2.5.8. Unreserved Bandwidth

The Unreserved Bandwidth sub-TLV specifies the amount of bandwidth

not yet reserved at each of the eight priority levels, in IEEE floating point format. The values correspond to the bandwidth that can be reserved with a setup priority of 0 through 7, arranged in increasing order with priority 0 occurring at the start of the sub-TLV, and priority 7 at the end of the sub-TLV. The initial values (before any bandwidth is reserved) are all set to the Maximum Reservable Bandwidth. Each value will be less than or equal to the Maximum Reservable Bandwidth. The units are bytes per second.

The Unreserved Bandwidth sub-TLV is TLV type 8, and is 32 octets in length.

2.5.9. Administrative Group

The Administrative Group sub-TLV contains a 4-octet bit mask assigned by the network administrator. Each set bit corresponds to one administrative group assigned to the interface. A link may belong to multiple groups.

By convention the least significant bit is referred to as 'group 0', and the most significant bit is referred to as 'group 31'.

The Administrative Group is also called Resource Class/Color [2].

The Administrative Group sub-TLV is TLV type 9, and is four octets in length.

3. Elements of Procedure

Routers shall originate Traffic Engineering LSAs whenever the LSA contents change, and whenever otherwise required by OSPF (an LSA refresh, for example). Note that this does not mean that every

change must be flooded immediately; an implementation MAY set thresholds (for example, a bandwidth change threshold) that trigger immediate flooding, and initiate flooding of other changes after a short time interval. In any case, the origination of Traffic Engineering LSAs SHOULD be rate-limited to at most one every MinLSInterval [1].

Upon receipt of a changed Traffic Engineering LSA or Network LSA (since these are used in traffic engineering calculations), the router should update its traffic engineering database. No SPF or other route calculations are necessary.

4. Compatibility Issues

There should be no interoperability issues with routers that do not implement these extensions, as the Opaque LSAs will be silently ignored.

The result of having routers that do not implement these extensions is that the traffic engineering topology will be missing pieces; however, if the topology is connected, TE paths can still be calculated and ought to work.

5. Normative References

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

[4] Kompella, K., Rekhter, Y., et al, "OSPF Extensions in Support of Generalized MPLS," work in progress.

[6] Kompella, K., and Y. Rekhter, "Signalling Unnumbered Links in RSVP-TE," work in progress.

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

[8] Coltun, R., "The OSPF Opaque LSA Option," RFC 2370, July 1998.

[9] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", Standard 754-1985, 1985 (ISBN 1-5593-7653-8).

6. Informative References

[2] Awduche, D., et al, "Requirements for Traffic Engineering Over MPLS," RFC 2702, September 1999.

[3] Smit, H. and T. Li, "ISIS Extensions for Traffic Engineering," work in progress.

[5] Kompella, K., Rekhter, Y., and A. Kullberg, "Signalling Unnumbered Links in CR-LDP," work in progress.

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

[11] Murphy, S., Badger, M., and B. Wellington, "OSPF with Digital Signatures", RFC 2154, June 1997.

7. Security Considerations

This document specifies the contents of Opaque LSAs in OSPFv2. As Opaque LSAs are not used for SPF computation or normal routing, the extensions specified here have no affect on IP routing. Tampering with TE LSAs may have an effect on traffic engineering computations, however, and it is suggested that whatever mechanisms are used for securing the transmission of normal OSPF LSAs be applied equally to all Opaque LSAs, including the TE LSAs specified here.

Note that the mechanisms in [1] and [11] apply to Opaque LSAs. It is suggested that any future mechanisms proposed to secure/ authenticate OSPFv2 LSA exchanges be made general enough to be used with Opaque LSAs.

8. IANA Considerations

The top level Types in a TE LSA as well as Types for sub-TLVs in a TE Link TLV are to be registered with IANA.

Following the guidelines set in [10], top level Types in TE LSAs from

3 through 32767 are to be assigned by Expert Review (the said Expert
to be decided by the IESG). Types from 32768 through 65535 are reserved for Private Use. In all cases, assigned values Types MUST be registered with IANA.

Also, sub-Types of a TE Link TLV from 10 to 32767 are to be assigned
by Expert Review; values from 32768 through 32772 are reserved for Private Use; and values from 32773 through 65535 are to be assigned

First Come First Served. In all cases, assigned values are to be registered with IANA.

9. Authors' Addresses

Dave Katz
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089 USA

Phone: +1 408 745 2000
Email: dkatz@juniper.net

Derek M. Yeung
Procket Networks, Inc.
1100 Cadillac Court
Milpitas, CA 95035 USA

Phone: +1 408 635-7900
Email: myeung@procket.com

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089 USA

Phone: +1 408 745 2000
Email: kireeti@juniper.net

10. IPR Notices

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any

copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

11. Full Copyright Notice

Internet Draft
Document: draft-schelen-nsis-opopsig-00.txt
Alban Couturier
Alcatel

Olov Schelen

Operax
Expires: December 2002

June 2002

On path and off path signaling for NSIS
<draft-schelen-nsis-opopsig-00.txt>

Status of this Memo

Abstract

The NSIS Work group will develop the requirements, architecture and protocols for the next IETF steps on signaling. Two approaches about

the
signaling model have been discussed: on-path and off-path. This draft provides a conceptual comparison between on-path and off-path signaling
together with reasons for why an NSIS protocol should be designed to support both cases.
The collection of data objects to be carried by the protocol should basically be the same in both cases and will evolve over time as new usages for NSIS protocol are identified. The differences between the two flavors of this NSIS protocol are then explained.

Table of Contents

10

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

1. Introduction

The Next Steps in Signaling (NSIS) working group is chartered to develop the requirements, architecture and protocols for the next IETF steps on signaling. Two approaches about the signaling model have been discussed: on-path and off-path. This draft provides a conceptual comparison between those two approaches together with reasons for why an NSIS protocol should be designed to support them both by offering two different flavors.
The collection of data objects to be carried by the protocol should basically be the same in both cases and will evolve over time as new usages for NSIS are identified. We identify at a high level the differences between the on-path and off-path flavors of the protocol.

2. Terminology

QoS Initiator (QI): NSIS entity responsible for generating the QSCs for traffic flow(s) based on user or application requirements and signaling them to the network as well as invoking local QoS provisioning
mechanisms. This can be located in the end system, but may reside elsewhere in the network.[2]

QoS Controller (QC): NSIS entity responsible for interpreting the signaling carrying the user QoS parameters, optionally inserting/ modifying the parameters according to local network QoS management

policy, and invoking local QoS provisioning mechanisms. Note that the QoS controller might have very different functionality depending on where in the network and in what environment they are implemented.[2]

QoS Service Classes (QSC): Specification of the QoS requirements of a traffic flow or aggregate. Can be further sub-divided into user specific and network related parameters. [2]

3. On-path signaling

3.1. Description of on-path signaling

On-path signaling refers to the situation where signaling is bound to the data path of the IP flow to be signaled. This is typically the case
of RSVP [3]. The advantage of on-path signaling is that the equipments along the data path can receive configuration data just by receiving and forwarding signaling packets, following the traditional routing mechanisms. In case the equipments to be configured are along the path,
this mechanism is considered the simplest. It does not require additional configuration, as the signaling hops are the same as the IP forwarding ones.

3.2. Issues of on-path signaling

3.2.1 Signaling unaware clouds and signaling continuity

The on-path signaling relies on the fundamental assumption that packets
are all routed the same way, based on the IP destination address. This will be called in this draft the "traditional IP routing". Traditional IP routing is of course predominant in the Internet today, therefore on-path signaling benefits from deployment advantages. In case of an IP
cloud which is signaling unaware (e.g. because of over-provisioning), the signal can go through the cloud transparently, like a normal IP packet. At the egress router of that cloud, the signaling still follows
the data path and can be processed by the next hops. Even if an IP cloud following the traditional IP routing is signaling unaware, it is still an on-path signaling carrier.

This situation may suffer from threats in several deployments. QoS routing, traffic engineering or load balancing technologies may route differently flows than in the traditional IP routing model. In these cases, a signaling unaware cloud is not anymore a signaling carrier, as nothing assures it will forward the signal at the same place it will
forward the dataflow. Then, signaling unaware clouds can break the on-path signaling, and can simply introduce bad state installations downstream.

3.2.2 Integration with policy servers

An issue is raised when states must be installed or processed by hosts that are not located on the data path. This happens for example when a server hosting user profiles participates in the admission control procedure. The current solution is to outsource the signaling processing via the COPS interface [4]. As policies are more and more important especially in case of peering between several ISPs, policy servers are expected to be extensively involved in signaling processing.

The on-path model does not easily provide a natural interface between domain boundaries (e.g., client-provider, provider-provider). Accountable services (e.g., max bandwidth with ensured QoS, max number of concurrent calls, variations over time of day and day of week, advance reservations, price profiles) require installing states in policy servers that are located off-path. This typically requires interaction between client and QoS controller (policy server) that is not easily supported by on-path signaling.

3.2.3 Separation of signaling processing and routers

One problem of RSVP and intserv is that every router in the network must
implement intserv and RSVP in order to link the sink and the source with a chain of QoS capable routers that all allocate resources for the
flow. Of course, RSVP aggregation [5], or RSVP over diffserv [6] [7] architectures propose to reduce the number of flow states inside network domains by concentrating per-flow reservations in strategically
positioned routers, called RSVP aggregators and de-aggregators (e.g., positioned at diffserv ingress and egress routers). However, this model
still imposes that all flow-aware nodes must implement the signaling. In a network consisting of several providers, de-aggregation into

micro-flows and re-aggregation is performed at all domain boundaries (ingress points and peering points). This imposes that a large portion of the routers must be upgraded in order to build a coherent signaling and QoS capable network.

4. Off-path signaling

4.1. Description of Off-Path signaling

Off-path signaling refers to signaling which does not follow the IP flows to be signaled. This happens either when signaling is not initiated by end hosts, or when signaling is directed to QoS controllers
that are not on the data path. In this model, the IP destination address of the signaling is separated from the destination address flow(s) for which resources are requested.

4.2. Benefits of Off-path signaling

4.2.1 Independence of signaling plane with forwarding plane

By nature, off-path signaling isolates signaling processing (e.g., admission control in case of QoS signaling) from the underlying network
nodes.
Because the complexity of the service control and admission control is isolated in servers, it allows in the short term to implement QoS signaling on top of a simple diffServ network. The advantage is to preserve the IP legacy of stateless class-based forwarding (not requiring state with respect to individual data flows). This provides scalability in routers, both in control and forwarding plane. Signaling
can be carried out at the application layer between QoS initiators and QoS controllers. Therefore, upgrading routers to a new signaling standard is not necessary to support off-path signaling.
Off-path signaling offers the same type of benefits in the long term. The separation between the signaling layer and the IP forwarding layer allows an ISP to isolate functionalities, and make them evolve

independently. There is a better flexibility in network evolution as new routers or nodes can be integrated in the network without having to be per-flow or NSIS session aware. Also, new algorithms for admission control can be deployed without having to upgrade the routers.

4.2.2 Flexibility in signaling entity placement

VoIP or multimedia network applications relies on servers such as soft-
switches, gatekeepers, SIP proxies and application servers that are not
on the data path of the multimedia flows. These servers are candidates to be QI, as they are responsible for the service sessions. These kind of QIs could use an off-path signalling protocol to interact with a QC.
Possible alternatives could be to deploy NSIS stacks and QSC aware applications in every VoIP/multimedia terminal along with synchronization with the network applications based on signaling outsourcing , or to insert on-path signaling proxies. These solutions require complex deployments that can be avoided thanks to the placement's flexibility associated to the off-path model.

4.2.3 Support for non-traditional routing

As seen in the on-path signaling section, a network based on non-traditional routing, because of e.g. traffic-engineering, may deviate the signaling from the flow to be signaled, and thus may not be a signaling carrier. Therefore, a network based on non-traditional IP routing must implement a signaling carrier function in order to be a decent transit network between QI and QR. This function can be implemented in the routers, or in off-path QoS controllers. In the later case, alternative routing rules that are implemented in the network must be replicated in the QoS controllers.

4.3. Issues of off-path signaling

4.3.1 Synchronization with routing tables

In addition to admission control or state installation, an off-path

QoS
controller must determine where to route the signaling, depending on where the flow to be signaled is routed. In case of on-path signaling, the signaling packet is ignored or processed, and then routed as a normal packet.
An off-path QoS-controller must determine where particular traffic leaves its domain and enters a neighboring domain. For this, topology awareness is needed. This draft does not intend to give an exhaustive list of architectures enabling a routing synchronization between the forwarding plane, and the signaling plane as this is out of the scope of the NSIS charter. However, there are solutions that can be implemented by passively participating in intra-domain routing (e.g., OSPF, IS-IS) and listening to inter-domain routing (e.g., by IBGP to edge routers inside the domain). The functionality is similar to a router but passive in the sense that no routes are advertised and no peering is performed with routers in other domains.

4.3.2 Admission control

In case the QoS controller's customers are not trusted, off-path signaling may involve configuration of edge traffic conditioners in order to do policing at the edges. Efficient standard approaches should
be defined for this. Various standards and proprietary interfaces can be supported by a QoS controller. SNMP, or COPS are two IETF standards that can be used to interface with edge routers.
It has to be noted that remote configuration of network elements may have a performance issue.

5. A Mixed solution

Up to this point, the draft presents the advantages and issues of on-path and off-path signaling, as it has been discussed in the mailing
list. The aim of this chapter is to advocate for a non-exclusive solution.

5.1. On-path and Off-path models inter-working

A strategy for NSIS could be to focus on a particular model, the preferred one being on-path, and refuse or postpone the work concerning
off-path.
As both model have their benefits and weaknesses, depending on the environment, the NSIS WG solution should be flexible enough to allow them both. There are situations where the signaling models could be combined for the same flow.

For example, the following situation should be possible:

- One ISP may use an application-level off-path solution to provide services to its customers, while continuing the signaling on-path towards a peering domains that adopt traditional IP routing. This situation corresponds to the gatekeeper/SIP proxy/content server initiating a reservation.

- One ISP may prefer on-path signaling from terminals and at access router link forwards the request to a policy framework that can take decisions based on customer profiles and network status, and also based
on contracts with neighbouring domains using off-path signaling.

- At the border between a domain following the traditional IP routing with another domain which adopts e.g. traffic engineering techniques, the on-path signaling can be extracted and then continued off-path.

These situations need a "signaling gateway router" implementing on-path
signaling on some of its interfaces and off-path signaling on the others.

Because it would be useful to have a simple implementation of the signaling gateway router, and because the additional required specification work is small, a unified solution presenting two flavors - on-path and off-path - of the same signaling is a reasonable choice. The following sections will explain the differences between these two flavors.

5.2. Data objects for on-path and Off-path signaling

5.2.1 Destination address

The information that are used to identify a flow are generally port numbers and IP address/prefix for destination and origination, protocol
number, DSCP/TOS value and, in IPv6, flow label. In case of on-path signaling for a micro-flow, the destination address of the flow must be

the same than the one of the signaling packet. However, this does not preclude a replication of the IP destination address in the IP payload of the signaling packet. This is the case in RSVP. When the flow is an aggregate, there must be in the signaling packet's header an IP destination address chosen inside the aggregate prefix, and the prefix itself must be inside the signaling packet payload.
So, concerning on-path model, the signaling carries a flow specification
that can contain a destination address or prefix.

In the off-path case, the signaling carries a flow specification that must always contain a destination address or prefix, as the packet header contains in destination address the IP address of the next QoS controller.

5.2.2 Ingress address

In the off-path model, it is necessary to specify in the request what is the entry point of a flow in the domain controlled by a QC. This can
be done by using the IP address of the router interface that receives the flow or alternatively the source address for a host sending a request to its local ISP (provided it is well-defined which ingress interface that the host will use). The reason for this is that with off-path signaling, the requests sent to a QoS controller can encompass
flows entering the domain through several interfaces of one or several routers. In order to know which router and which interface will receive
the flow, this information must be added in the request. For requests between adjacent QoS controllers the upstream QoS controller must find out which incoming interface of the downstream domain that will be used.

To summarize, the differences between an on-path signaling and its associated off-path version are:
- the addition, if not already mandatory in on-path, of the IP destination address/prefix in the flow specification
- the addition of an ingress address object in the flow specification.

It is expected that a signaling gateway router receiving an on-path signaling message, after having processed it, will add the destination address (if needed) and the ingress address in the message, and forward
it.

5.3. Protocol concepts

On-path signaling has traditionally been implemented as a specific protocol on top of IP that is interpreted by routers along the path. It is likely that an NSIS on-path flavor will be designed along these lines as routers typically are not involved in application layer signaling. This is a quite complex task both in terms of specification and deployment in routers.

An off-path signaling flavor can be implemented at the application layer
(over TCP, UDP or other transport protocol). The design can therefore focus more on specifying data-objects as no new support is needed for transport functionality. It will also be possible to try out and deploy
the signaling in networks with current diffserv standard, without requiring new standards in the routers.

Both on-path and off-path models should use pair-wise handshake between
QoS controllers involved in providing e2e service. Early on-path protocols (e.g., RSVP) did signal along the path without handshake for reliable delivery between adjacent neighbors, but it has been identified [8] that such a model has problems meeting required state maintenance.

6. Conclusion

Both on-path and off-path models are relevant for signaling in IP networks, and answer technical needs. In order to increase the applicability and deployment of a new signaling, this draft proposes to specify in NSIS one protocol that has one on-path and one off-path flavor. The identified differences between the two variants are one or

two protocol objects defining ingress and destination address for requests. The off-path flavor may be implemented at application layer, while the on-path flavor most likely would be implemented as a protocol
on top of IP.

7. Security Considerations

TBC

8. References

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

[2] Brunner, M., "Requirements for QoS Signaling Protocols", draft-ietf-nsis-req-02.txt, May 2002, Work in progress

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

[4] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja, R. and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[5] Baker F., Iturralde C., Le Faucheur F., Davie B., "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, Septembe 2001.

[6] Black, D., Blake, S., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[7] Bernet Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L., Speer, M.,
Braden, R., Davie, B., Wroclawski, J. and E. Felstaine, "Integrated Services Operation Over Diffserv Networks", RFC 2998, November 2000.

[8] Braden, R., Lindell, B., "A Two-level Architecture for Internet Signalling" draft-braden-2level-signal-arch-00.txt, Nov 2001.

9. Author's Addresses

Alban Couturier
Ets de Marcoussis R&I/ULC
Route de Nozay
91461 Marcoussis CEDEX, France
Email: Alban.Couturier@alcatel.fr

Olov Schelen
Operax AB
Aurorum 8
SE 97775 Lulea, Sweden
Email: Olov.Schelen@operax.com

10. Full Copyright Statement

NSIS O. Schelen
Internet-Draft
Operax
Expires: May 5, 2003 A. Couturier

Alcatel
R. Bless

Univ. Karlsruhe
R. Geib

T-Systems
O. Dugeon

FTR&D
November 4, 2002

Path-coupled and Path-decoupled Signaling for NSIS
draft-schelen-nsis-opopsig-01

Status of this Memo

http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 5, 2003.

Abstract

The NSIS Work group will develop the requirements, architecture and protocols for the next IETF steps on signaling. Two approaches for a signaling model have been discussed: path-coupled and path-decoupled (previously denoted as on-path and off-path). This draft provides a

conceptual comparison between path-coupled and path-decoupled signaling together with reasons for why an NSIS protocol should be designed to support both cases. The collection of data objects to be carried by the protocol should basically be the same in both cases and will evolve over time as new usages for NSIS protocol are identified. The differences between the two flavors of this NSIS protocol are then explained.

1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

2. Introduction

The Next Steps in Signaling (NSIS) working group is chartered to develop the requirements, architecture and protocols for the next IETF steps on signaling. Two approaches about the signaling model have been discussed: path-coupled and path-decoupled. This draft

provides a conceptual comparison between those two approaches together with reasons for why an NSIS protocol should be designed to support them both by offering two different flavors. The collection of data objects to be carried by the protocol should basically be the same in both cases and will evolve over time as new usages for NSIS are identified. We identify at a high level the differences between the path-coupled and path-decoupled flavors of the protocol.

3. Terminology

NSIS Initiator (NI) - NSIS Entity that initiates NSIS signaling for a network resource based on user or application requirements. This can be located in the end system, but may reside elsewhere in network [2].

NSIS Responder (NR) - NSIS Entity that terminates NSIS signaling and can optionally interact with applications as well.

NSIS Forwarder (NF) - NSIS Entity on the path between a NI and NR, which may interact with local resource management function (RMF) for this purpose. NSIS Forwarder also propagates NSIS signaling further through the network. It is responsible for interpreting the signaling carrying the user parameters, optionally inserting or modifying the parameters according to domain network management policy [2].

Control Information: information that governs the treatment to be applied to a flow or an aggregate (e.g., QoS treatment including

service class, flow administration, and any associated security or accounting information [2].

4. Path-coupled Signaling

4.1 Description of path-coupled signaling

Path-coupled signaling refers to the situation where the signaling path is tied to the data path of the IP flow to be signaled. That means signaling messages referring to a particular data flow follow the same path as packets of that data flow, i.e., signaling messages pass through the same devices as data packets of the data flow.

In the forward direction signaling messages usually carry the same IP destination address as data packets of the related user data flow. It must be noted, that some signaling messages (e.g., responses) may also follow the same path in the reverse direction. Usually, the previous hop must be remembered and directly addressed in this case.
An example for a path-coupled signaling protocol is RSVP [4].

In summary path-coupled signaling shows the following properties:

- Only network elements that forward data packets can participate in signaling, i.e., routers must process the signaling messages.

- Routers that forward IP packets along the data path can participate in the signaling by intercepting and processing incoming signaling messages.

- The next signaling hop in forwarding direction is discovered by transmitting signaling packets through an ordinary lookup in the local IP forwarding table, using the final destination address of the signaling packets. The signaling path in the reverse direction (upstream) has to be remembered from the forward direction (i.e., the previous hop must be stored), because routes can be asymmetric.

- The path of the signaling messages adapts automatically to route changes for data packets (if associated with a soft-state mechanism).

- Signaling unaware routers can forward a signaling message correctly to the next hop if the destination address of the signaling message is the same as for the data flow.

4.2 Advantages of path-coupled signaling

4.2.1 Efficient device configuration

One advantage of path-coupled signaling is that nodes along the data path can install or update configuration data (state) just by receiving, processing and forwarding signaling packets, following the traditional routing mechanisms. In case the devices to be configured are located along the path, this mechanism is considered the simplest. It does not require additional configuration protocol exchange, as the signaling hops are the same as the IP forwarding ones.

4.2.2 Bypassing signaling unaware clouds

In case of an IP cloud which is signaling unaware (e.g., because of over-provisioning), signaling packets can go through the cloud transparently, like normal IP packets. At the egress router of that cloud, signaling still follows the data path and can be processed by the next hops. Even if an IP cloud following the traditional IP routing is signaling unaware, it is still a path-coupled signaling carrier.

4.2.3 Automatic adaptation to changed routes

The path of signaling messages adapts automatically to route changes which is often, but not always, desired. As a result of route changes, reservation state will automatically be installed in routers along a new path while it will be removed in routers along the old

path.

4.3 Disadvantages of path-coupled signaling

4.3.1 Limited flexibility for integration of other entities

Typically IP networks are provisioned for delivering certain services internally, while customers/peers have various access schemes at edges/boundaries. Accountable services (e.g., max bandwidth with ensured QoS, max number of concurrent VoIP calls, access bandwidth variations over time of day and day of week, advance reservations, price profiles) require installing states in policy servers that are not located on the data path. Establishing such services typically requires interaction between client and NSIS forwarder (policy server) to perform state processing and state installation.

However, entities and hosts that are not located on the data path cannot be easily included into a path-coupled signaling process. This makes it more difficult to use signaling proxies or

administrative servers (e.g., policy or accounting servers). The latter is for example required when a server hosting user profiles participates in the admission control procedure. In general, all actions that require keeping persistent states, e.g., for accounting or retrieving service level data (such as user profiles or policies) cannot be easily supported by routers. The current solution is to out-source the signaling processing via the COPS interface [5]. As policies are more and more important especially in cases of customer access and peering between several ISPs, policy servers are expected to be extensively involved in signaling processing.

4.3.2 Impact of non contiguous signaling paths

Path-coupled signaling is based on the fundamental assumption that

the signaling path is the same as the data path. Usually, in a stable network condition (no route changes occur) consecutive packets of a single flow are all routed the same way, based on the IP destination address. This will be called "traditional IP routing" in the following. Traditional IP routing is of course predominant in the Internet today, therefore path-coupled signaling benefits from deployment advantages.

This situation may suffer from threats in several deployments. QoS routing, traffic engineering or load balancing technologies may route flows differently than in the traditional IP routing model. In these cases, a signaling unaware cloud is not anymore a transparent signaling carrier, as nothing assures it will forward the signal at the same place it will forward the data flow. Then, signaling unaware clouds can break the path-coupled signaling, and can simply install reservation state at wrong paths.

4.3.3 Signaling processing and complex control functions in routers

One problem is that signaling message processing and more complex control tasks (e.g., resource-based admission control) have to be implemented in routers. A change of control functions (e.g., admission control algorithms) requires also a change to the routers.

One problem of RSVP and IntServ is that every router in the network must implement IntServ and RSVP in order to link the sink and the source with a chain of QoS capable routers that all allocate resources for the flow. Of course, RSVP aggregation [6], or RSVP over DiffServ [7][8] architectures propose to reduce the number of flow states inside network domains by concentrating per-flow reservations in strategically positioned routers, called RSVP aggregators and de-aggregators (e.g., positioned at diffserv ingress and egress routers). However, this model still imposes that all flow-aware nodes must implement the signaling. In a network

consisting of several providers, de-aggregation into micro-flows and re-aggregation is performed at all domain boundaries (ingress points and peering points). This imposes that a large portion of the routers must be upgraded in order to build a coherent signaling and QoS capable network.

4.3.4 Limited support in mobile scenarios

In some scenarios with mobile senders or receivers it may be desirable to have a "seamless" hand-over. In this case, resources along the new path should be reserved before the data flow is actually switched from the old path to the new path.

4.3.5 Protection and Security

Path-coupled signaling messages being transmitted to an unknown next signaling hop may be hard to protect.

4.3.6 NAT and private address schemes

Path-coupled signaling messages transmitted through a Firewall/NAT must be changed when passing this device. When a network operator uses a private address scheme, the end user IP address must be translated before reaching the public part of the network. NAT devices that translate addresses in headers must also translate addresses carried in the body of signaling messages to reflect the NAT processing. To achieved this goal, NAT devices must be NF devices to convey NSIS compliant signaling messages.

5. Path-decoupled Signaling

5.1 Description of Path-Decoupled Signaling

Path-decoupled signaling refers to the situation where the signaling path is not necessarily bound to the data path of the signaled flow. End stations/users may signal to particular entities (e.g., servers) in the network of their providers. The "path" taken by path-decoupled signaling messages correspond to the AS path rather than the hop by hop path taken by path-coupled signaling messages. That means signaling messages may be destined to devices that are not on the forwarding path of the particular flow. This happens either when signaling is not initiated by end hosts, or when signaling is

directed to NSIS forwarders that are not on the data path. In this model, the IP destination address of a signaling message is separated from the destination address flow(s) for which resources are requested.

In summary path-decoupled signaling shows the following properties:

- The signaling path is decoupled from the actual data path, therefore it allows to signal entities that are not on the data path. This allows to shift some control functions to other entities than routers.
- Path-decoupled signaling simplifies interworking with domains applying forwarding planes other than IP (e.g. MPLS or WDM) or using private addressing schemes.
- To modify and control resources at the routers passed by the data-flow corresponding to a path-decoupled signaling message, the data path must be predicted by path-decoupled signaling units.
- The path-decoupled signaling system must be able to configure routers in the data path by access to a management interface.
- Path-decoupled signaling must be able to discover the next signaling hop.

5.2 Advantages of path-decoupled signaling

5.2.1 Independence of signaling plane and forwarding plane

By nature, path-decoupled signaling isolates signaling processing (e.g., admission control in case of QoS signaling) from the underlying network nodes. Because the complexity of the service control and admission control is isolated in servers, it allows in the short term to implement QoS signaling on top of a simple DiffServ

network. A similar architecture based on pre-provisioned networks is explained in [10]. The advantage is to preserve the IP legacy of stateless class-based forwarding (not requiring state with respect to individual data flows). This provides scalability in routers, both in control and forwarding plane. Signaling can be carried out at the application layer between NSIS initiators and NSIS Forwarders. Path-decoupled signaling offers the same type of benefits in the long term. The separation between the signaling layer and the IP forwarding layer allows an ISP to isolate functionalities, and make them evolve independently. There is flexibility in network evolution as new routers or nodes can be integrated in the network without having to be per-flow or NSIS session aware. Also, new algorithms for admission control can be deployed without having to upgrade the routers.

5.2.2 Low upgrade complexity in routers

To support a path-decoupled signaling standard, upgrading of routers is not needed or may be limited to a relay function identifying an

NSIS message and transmitting it to a path-decoupled signaling unit responsible for this router. The latter method can be used for interworking with path-coupled signaling.

5.2.3 Flexibility in signaling entity deployment

In deployment of a new protocol, there may be some applications and end-points that are NSIS aware while others are not. The path-decoupled model can work transparently to end-points and application by having NSIS initiated from application frameworks or from separate network/resource management frameworks. Also, endpoints or web servers can offer applications allowing end-users to self-manage

their general purpose network access. VoIP or multimedia network applications relies on servers such as soft- switches, gatekeepers, SIP proxies and application servers that are not on the data path of the multimedia flows. These servers are candidates to be an NI, as they are responsible for the service sessions. These kinds of NIs could use a path-decoupled signaling protocol to interact with an NF.

5.2.4 Support for non-traditional routing

Network sections applying forwarding planes other than IP may require an interworking functionality with NSIS signaling. While layer 2 issues are out of scope for NSIS, MPLS is operational in several large international IP backbones. While a path-coupled signaling architecture requires an IP/MPLS gateway to implement the new NSIS protocol, an interworking function and possibly some modified MPLS signaling protocol, a path-decoupled system could take care of all that. End to end service deployment across heterogeneous network platforms may benefit from path-decoupled signaling.

5.2.5 Mobility

Path-decoupled signaling enables seamless hand-overs combined with a reduction of signaling in the case of wireless mobility. A path-decoupled signaling unit learning about a mobile terminal now connected to a new access router may transfer the signaling context of the mobile terminal to the new access router and simultaneously remove state in the old access router. No additional air interface signaling is required to re-install state in the new access router. The resulting hand-over is fast and saves scarce battery power.

5.2.6 Security

Path-decoupled signaling units may discover neighboring path-decoupled signaling units prior to any end to end service reservation. Hence, it is sound to expect signaling between path-decoupled systems to be protected once end to end messages are

processed. Similar to security mechanisms to be applied by NSIS, the discovery mechanism for a path-decoupled protocol may not have to be specified by NSIS.

5.3 Disadvantages of path-decoupled signaling

5.3.1 Determining the next signaling hop

Because signaling entities are not placed along the data path, the next destination of a signaling message cannot be determined by determining the next hop in the data path.

However, there are several ways to determine the next signaling node. One possibility is to extend legacy configuration mechanisms at access networks such as DHCP or stateless auto configuration by the required addresses of NFs. Adjacent domains may have either statically configured next hops or may use an extra discovery mechanism. Routing tables can be used to determine which domain is the next hop.

5.3.2 Synchronization with routing tables

In addition to admission control or state installation, a path-decoupled NSIS Forwarder must determine where to route the signaling messages, depending on where the flow to be signaled is routed. In case of path-coupled signaling, the signaling packet is ignored or processed, and then routed as a normal packet.

A path-decoupled NF must determine where particular traffic leaves its domain and enters a neighboring domain. For this, topology awareness is needed. This draft does not intend to give an exhaustive list of architectures enabling a routing synchronization between the forwarding plane, and the signaling plane as this is out of the scope of the NSIS charter. However, there are solutions that can be implemented by passively participating in intra-domain routing (e.g., OSPF, IS-IS) and listening to inter-domain routing (e.g., by IBGP to edge routers inside the domain). The functionality is similar to what is found inside a router today but passive in the sense that no routes are advertised and no peering is performed with routers in other domains.

5.3.3 Installing state in routers

When considering QoS provisioning in DiffServ networks, path-decoupled signaling typically involves configuration of traffic conditioners at domain boundaries in order to perform policing and marking at the network edges (especially in the very first router). Various standards and proprietary interfaces can be supported by an

NSIS Forwarder in order to transport the necessary configuration data (e.g., profiles) to the routers. SNMP or COPS are two IETF standards that can be used to configure routers. It has to be noted that this signaled configuration of network elements may have a performance issue due to the available mechanisms.

6. A Combined Solution

Up to this point, the draft presents the advantages and issues of path-coupled and path-decoupled signaling, as it has been discussed in the mailing list. The aim of this chapter is to advocate for a non-exclusive solution.

6.1 Path-coupled and path-decoupled models inter-working

A strategy for NSIS could be to focus on a particular model, the preferred one being path-coupled, and refuse or postpone the work concerning path-decoupled. As both models have their benefits and weaknesses, depending on the environment, the NSIS WG solution should be flexible enough to allow them both. There are situations where the signaling models could be combined for the same flow.

For example, the following situation should be possible:

- One ISP may use an application-level path-decoupled solution to provide services to its customers, while continuing the signaling path-coupled towards a peering domains that adopt traditional IP

routing. This situation corresponds to the gatekeeper/SIP proxy/
content server initiating a reservation.

- One ISP may prefer path-coupled signaling from terminals and at access router link forwards the request to a policy framework that can take decisions based on customer profiles and network status, and also based on contracts with neighboring domains using path-decoupled signaling.

- At the border between a domain following the traditional IP routing with another domain which adopts e.g. traffic engineering techniques, the path-coupled signaling can be extracted and then continued path-decoupled.

These situations need a "signaling gateway router" implementing path-coupled signaling on some of its interfaces and path-decoupled signaling on the others.

Because it would be useful to have a simple implementation of the signaling gateway router, and because the additional required specification work is small, a unified solution presenting two

flavors - path-coupled and path-decoupled - of the same signaling is a reasonable choice. The following sections will explain the differences between these two flavors.

6.2 Data objects for path-coupled and path-decoupled signaling

6.2.1 Destination address

The information that are used to identify a flow are generally port numbers and IP address/prefix for destination and origination, protocol number, DSCP/TOS value and, in IPv6, flow label. In case of

path-coupled signaling for a micro-flow, the destination address of the flow must be the same as the one of the signaling packet. However, this does not preclude a replication of the IP destination address in the IP payload of the signaling packet. This is the case in RSVP. When the flow is an aggregate, there must be in the signaling packet's header an IP destination address chosen inside the aggregate prefix, and the prefix itself must be inside the signaling packet payload. So, concerning path-coupled model, the signaling carries a flow specification that can contain a destination address or prefix.

In the path-decoupled case, the signaling carries a flow specification that must always contain a destination address or prefix, as the packet header contains the destination address of the next NF.

6.2.2 Ingress address

In the path-decoupled model, it is necessary to specify in the request what is the entry point of a flow in the domain controlled by an NF. This can be done by using the IP address of the router interface that receives the flow or alternatively the source address for a host sending a request to its local ISP (provided it is well-defined which ingress interface that the host will use). The reason for this is that with path-decoupled signaling, the requests sent to an NF can encompass flows entering the domain through several interfaces of one or several routers. In order to know which router and which interface will receive the flow, this information must be added in the request. For requests between adjacent NFs the upstream NFs must find out which incoming interface of the downstream domain that will be used.

To summarize, the differences between an path-coupled signaling and its associated path-decoupled version are:

- the addition, if not already mandatory in path-coupled, of the IP destination address/prefix in the flow specification

- the addition of an ingress address object in the flow specification.

It is expected that a signaling gateway router receiving an path-coupled signaling message, after having processed it, will add the destination address (if needed) and the ingress address in the message, and forward it.

6.3 Protocol concepts

Path-coupled signaling has traditionally been implemented as a specific protocol on top of IP that is interpreted by routers along the path. It is likely that an NSIS path-coupled flavor will be designed along these lines as routers typically are not involved in application layer signaling. This is a quite complex task both in terms of specification and deployment in routers.

A path-decoupled signaling flavor can be implemented at the application layer (over TCP, UDP or other transport protocol). The design can therefore focus more on specifying data-objects as no new support is needed for transport functionality. It will also be possible to try out and deploy the signaling in networks with current diffserv standard, without requiring new standards in the routers.

Both path-coupled and path-decoupled models should use pair-wise handshake between NFs involved in providing e2e service. Early path-coupled protocols (e.g., RSVP) did signal along the path without handshake for reliable delivery between adjacent neighbors, but it has been identified [9] that such a model has problems meeting required state maintenance.

7. Conclusion

Both path-coupled and path-decoupled models are relevant for signaling in IP networks, and answer technical needs. In order to increase the applicability and deployment of a new signaling, this document proposes to specify in NSIS one protocol that has one path-

coupled and one path-decoupled flavor. The identified differences between the two variants are one or two protocol objects defining ingress and destination address for requests. The path-decoupled flavor may be implemented at application layer, while the path-coupled flavor most likely would be implemented as a protocol on top of IP.

8. Security Considerations

Because this document only discusses aspects of path-coupled and path-decoupled signaling there are no direct security implications.

However, for both signaling modes several security mechanisms should be applied, especially integrity protection and authentication of signaling messages in order to prevent unauthorized usage of resources and to allow proper accounting.

Thus, several security mechanisms can be applied and combined, e.g., using IPsec mechanisms to secure the transport of signaling messages, use of dedicated authentication and integrity protection mechanisms in the signaling protocol itself as well as integration of existing AAA solutions.

9. Acknowledgements

The terms path-coupled and path-decoupled were proposed by Robert Hancock et al. in [3].

References

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

[2] Brunner, M., "Requirements for Signaling Protocols", draft-ietf-nsis-req-04 (work in progress), August 2002.

[3] Freytsis, I., "Next Steps in Signaling: Framework", draft-ietf-nsis-fw-00 (work in progress), October 2002.

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

[5] Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R. and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[6] Baker, F., Iturralde, C., Le Faucheur, F. and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

[7] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[8] Bernet, Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J. and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000.

[9] Braden, B. and B. Lindell, "A Two-Level Architecture for Internet Signaling", draft-braden-2level-signal-arch-00 (work in progress), November 2001.

[10] De Clercq, J., Van den Bosch, S. and A. Couturier, "An architecture for a gradual deployment of end-to-end QoS on an Internet-wide scale", draft-declercq-vsn-arch-00 (work in progress), October 2002.

[11] Schulzrinne, H., Tschofenig, H., Fu, X., Eisl, J. and R. Hancock, "CASP - Cross-Application Signaling Protocol", draft-
schulzrinne-nsis-casp-00 (work in progress), September 2002.

Authors' Addresses

Olov Schelen
Operax AB
Aurorum 8
SE 97775 Lulea
Sweden

EMail: Olov.Schelen@operax.com
URI: http://www.operax.com

Alban Couturier
Alcatel
Ets de Marcoussis R&I/ULC
Route de Nozay
91461 Marcoussis CEDEX, France
FR

EMail: Alban.Couturier@alcatel.fr

Roland Bless
Institute of Telematics, Universitaet Karlsruhe (TH)
Zirkel 2
76128 Karlsruhe
Germany

Phone: +49 721 608 6413
EMail: bless@tm.uka.de
URI: http://www.tm.uka.de/~bless

Ruediger Geib
T-Systems Nova GmbH
Am Kavalleriesand 3
64295 Darmstadt
Germany

Phone: +49 6151 832 138
EMail: Ruediger.Geib@t-systems.com
URI: http://www.t-nova.de

Olivier Dugeon
France Telecom R&D
2 Avenue Pierre Marzin
F-22307 Lannion
France

Phone: +33 296 05 2880
EMail: Olivier.Dugeon@francetelecom.com

Full Copyright Statement

Acknowledgement

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