Network Working Group
Internet Draft
Expiration Date: February 1999

Loa Andersson
Bay Networks Inc.

Paul Doolan
Ennovate Networks

Nancy Feldman
IBM Corp

Andre Fredette
Bay Networks Inc.

Bob Thomas
Cisco Systems, Inc.

August 1998

LDP Specification

draft-ietf-mpls-ldp-01.txt

Status of this Memo

Abstract

An overview of Multi Protocol Label Switching (MPLS) is provided in [FRAMEWORK] and a proposed architecture in [ARCH]. A fundamental concept in MPLS is that two Label Switching Routers (LSRs) must agree on the meaning of the labels used to forward traffic between and

through them. This common understanding is achieved by using the Label Distribution Protocol (LDP) referenced in [FRAMEWORK] and [ARCH]. This document defines the LDP protocol.

Open Issues

The following LDP issues are left unresolved with this version of the spec:

- The loop prevention/detection mechanism to be employed by LDP. This spec has retained the path vector mechanism from previous drafts. However, draft-ohba-mpls-loop-prevention-01.txt has been proposed as an alternative.

- Support for explicitly routed LSPs. The need for this feature has been debated at length. This spec refines the previous version of the spec in this area. However, there remains some belief in the WG that explicitly routed LSPs should be supported by enhancements to RSVP and not LDP.

  The support for explicitly routed LSPs in the spec is independent of other LDP features and could, should the WG decide to do so, be removed without impact on other LDP features.

- Traffic engineering considerations beyond support for explicit routing.

- The need for all of the FEC types (called FEC elements in this version of the spec, SMDs in previous versions) is being debated. This version of the spec defines fewer FEC types than previous versions.

- LDP support for multicast is not defined in this version. Multicast support will be addressed in a future version.

- The message and TLV encodings are likely to change in some minor ways in the next draft of the spec.

Table of Contents

1. LDP Overview

LDP is the set of procedures and messages by which Label Switched Routers (LSRs) establish Label Switched Paths (LSPs) through a network by mapping network-layer routing information directly to data-link layer switched paths. These LSPs may have an endpoint at a directly attached neighbor (comparable to IP hop-by-hop forwarding), or may have an endpoint at a network egress node, enabling switching via all intermediary nodes.

LDP associates a forwarding equivalence class (FEC) [ARCH] with each LSP it creates. The FEC associated with an LSP specifies which packets are "mapped" to that LSP. LSPs are extended through a network as each LSR "splices" incoming labels for a FEC to the outgoing label assigned to the next hop for the given FEC.

Note that this document is written with respect to unicast routing only. Multicast will be addressed in a future revision.

Note that this document is written with respect to control-driven traffic. It describes mappings which are initiated for routes in the forwarding table, regardless of traffic over those routes. However, LDP does not preclude data-driven support.

1.1. LDP Peers

Two LSRs which use LDP to exchange label/stream mapping information are known as "LDP Peers" with respect to that information and we speak of there being an "LDP Session" between them. A single LDP adjacency allows each peer to learn the other's label mappings i.e. the protocol is bi-directional.

1.2. LDP Message Exchange

There are four categories of LDP messages:

1. Discovery messages, used to announce and maintain the presence of an LSR in a network.

2. Session messages, used to establish and maintain terminate sessions between LSR peers.

3. Advertisement messages, used to create, change, and delete label mappings for FECs.

4. Notification messages, used to provide advisory information and to signal errors.

Discovery messages provide a mechanism whereby LSRs continually indicate their presence in a network via the Hello message. This is transmitted as a UDP packet to the LDP port at the `all LSR routers' group multicast address. When an LSR chooses to establish a session with an LSR learned via the hello message, it uses the LDP initialization procedure over TCP transport. Upon successful completion of the initialization procedure, the two LSRs are LDP peers, and may exchange advertisement messages.

When to request a label or advertise a label mapping to a peer is largely a local decision made by an LSR. In general, the LSR requests a label mapping from a neighboring LSR when it needs one, and advertises a label mapping to a neighboring LSR when it wishes the neighbor to use a label.

Correct operation of LDP requires reliable and in order delivery of mappings (although there are circumstances when this second requirement could be relaxed). To satisfy these requirements LDP uses the TCP transport for adjacency, advertisement and notification messages.

1.3. LDP Error Handling

LDP errors and other events of interest are signaled to an LSR peer by notification messages.

There are two kinds of LDP notification messages:

1. Error notifications, used to signal fatal errors. If an LSR receives an error notification for an LDP session with a peer, it terminates the peer session by closing the TCP transport connection for the session and discarding all label mappings learned via the session.

2. Advisory notifications, used to pass an LSR information about the LDP session or the status of some previous message received from the peer.

1.4. LDP Extensibility and Future Compatibility

It is likely that functionality will be added to LDP after its initial release. It is also likely that this additional functionality will utilize new messages and object types (TLVs). It may be desirable to employ such new messages and TLVs within a network using older implementations that do not recognize them. While it is not possible to make every future enhancement backwards compatible, some prior planning can ease the introduction of new capabilities. This specification defines rules for handling unknown message types and unknown TLVs for this purpose.

2. LDP Operation

2.1. FEC Types

It is necessary to precisely define which IP packets may be mapped to each LSP. This is done by providing a FEC specification for each LSP. The FEC defines which IP packets may be mapped to the same LSP, using a unique label.

LDP supports LSP granularity ranging from end-to-end flows to the aggregation of all traffic through a common egress node; the choice of granularity is determined by the FEC choice.

Each FEC is specified as a list of one or more FEC elements. Each FEC element specifies a set of IP packets which may be mapped to the corresponding LSP.

Following are the currently defined types of FEC elements. New element types may be added as needed:

1. IP Address Prefix.

   This element provides a list of one or more IP address prefixes. Any IP packet whose destination address matches one or more of the specified prefixes may be forwarded using the associated LSP.

2. Router ID

   This element provides a Router ID (ie, a 32 bit IP address of a router). Any IP packet for which the path to the destination is known to traverse the specified router may be forwarded using the associated LSP. This element allows the full set of destinations reachable via a specified router to be indicated

in a single FEC element.

3. Flow

This element specifies a set of datagram information, such as port, dest-addr, src-addr, etc. This element provides LDP with the ability to support MPLS flows with no aggregation.

Where a packet maps to more than one FEC it is transmitted on the LSP associated with the FEC to which the packet has the 'most specific' match.

2.2. Mapping packets to FECs

FEC objects (TLVs) are transmitted in the LDP messages that deal with (advertise, request, release ad withdraw) FEC-Label mappings.

A stream of packets with a given destination network can be characterized by a single Address Prefix FEC Element. This results in each specified address prefix sustaining its own LSP tree. This singular mapping is recommended in environments where little or no aggregation information is provided by the routing protocols (such as within a simple IGP), or in networks where the number of destination prefixes is limited.

In environments where additional aggregation not provided by the routing protocols is desired, an aggregation list may be created. In this, all prefixes that are to share a common egress point may be advertised within the same FEC. This type of aggregation is configured.

The router ID FEC type may be used in any environment in which the routing protocols allow routers to determine the egress point for specific IP packets. For example, the router ID FEC type may be used in combination with BGP, OSPF, and/or IS-IS.

For example, the mapping between IP packets and the router ID may be provided via the BGP NEXT_HOP attribute.   When a BGP border LSR injects routes into the BGP mesh, it may use its own IP address or the address of its external BGP peer as the value of the NEXT_HOP attribute. If the BGP border ISR uses its own IP address as the NEXT_HOP attribute, then one LSP is created which terminates at the BGP border, and the border LSR will forward traffic at layer-3 towards its external BGP neighbors. If the BGP border LSR uses the external BGP peer as the NEXT_HOP attribute, then a separate LSP may be created for each external BGP neighbor, thereby allowing the border LSR to switch traffic directly to each of its external BGP

neighbors.

Similarly, the mapping between IP packet and router ID may be provided by OSPF. This is comprised of the Router ID of the router that initiated the link state advertisement. The Router ID may also be the OSPF Area Border Router.

Note that BGP and OSPF may share the same LSP when a given Router ID is found in both protocol's Routing Information Base.

The Router ID FEC allows aggregation of multiple IP address prefixes to the same LSP, without requiring that the prefixes be explicitly listed in the FEC. Also, it allows addresses advertised using OSPF and addresses advertised using BGP to be aggregated using the same LSP. Finally, when the set of addresses reachable via a router changes, and the changes are announced into the routing protocol (BGP, OSPF, and/or IS-IS), use of the routerID FEC eliminates the need to explicitly announce the route changes into LDP.

2.3. Label Spaces, Identifiers, Sessions and Transport

The notion of "label space" is useful for discussing the assignment and distribution of labels. There are two types of label spaces:

- Per interface label space. Interface-specific incoming labels are used for interfaces that use interface resources for labels. An example of such an interface is a label-controlled ATM interface which uses VCIs as labels, or a frame Relay interface which uses DLCIs as labels.

  Note that the use of a per interface label space only makes sense when the LDP peers are "directly connected" over an interface, and the label is only going to be used for traffic sent over that interface.

- Per platform label space. Platform-wide incoming labels are used for interfaces that can share the same labels.

An LDP identifier is a six octet quantity used to identify an LSR label space. The first four octets encode an IP address assigned to the LSR, and the last two octets identify a specific label space within the LSR. The last two octets of LDP Identifiers for platform-wide label spaces are always both zero. This document uses the following print representation for LDP Identifiers:

<IP address> : <Label space Id>

for example, 171.32.27.28:0, 192.0.3.5:2.

Note that an LSR that manages and advertises more than one label space uses a different LDP Identifier for each such label space.

A situation where an LSR would need to advertise more than one label space to a peer and hence use more than one LDP Identifier occurs when the LSR has two links to the peer and both are ATM (and use per interface labels). Another situation would be where the LSR had two links to the peer, one of which is ethernet (and uses per platform lables) and the other of which is ATM.

LDP sessions exist between LSRs to support label exchange between them.

When a LSR must use LDP to advertise more than one label space to another LSR it uses a separate LDP session for each label space rather than a single LDP session for all the label spaces.

LDP uses TCP as a reliable transport for sessions.

When multiple LDP sessions are required between two platforms there is one LDP session per TCP connection rather than many LDP sessions per TCP connection.

2.4. LDP Sessions between non-Directly Connected LSRs

LDP sessions between LSRs that are not directly connected at the link level may be desirable in some situations.

For example, consider a "traffic engineering" application where LSR LSR1 sends traffic matching some criteria via an LSP to non-directly connected LSR LSR2 rather than forwarding the traffic along its normally routed path.

An LDP session between LSR1 and LSR2 enables LSR2 to label switch traffic arriving on the LSP from LSR1. In this situation LSR1 applies two labels to traffic it forwards on the LSP. First, it adds the label learned via the LDP session with LSR2 to the packet label stack (either by replacing the label on top of the packet label stack with it if the packet arrives labeled or by pushing it if the packet arrives unlabeled). Next, it pushes the label for the LSP onto the label stack.

2.5. LDP Discovery

LDP discovery is a mechanism that enables an LSR to discover potential LDP peers. Discovery makes it unnecessary to explicitly configure an LSR's label switching peers.

There are two variants of the discovery mechanism:

- A basic discovery mechanism used to discover LSR neighbors that are directly connected at the link level.

- An extended discovery mechanism used to locate LSRs that are not directly connected at the link level.

2.5.1. Basic Discovery Mechanism

To engage in LDP Basic Discovery on an interface an LSR periodically sends LDP Link Hellos out the interface. LDP Link Hellos are sent as UDP packets addressed to the well known LDP discovery port for the "all routers" group multicast address.

An LDP Link Hello sent by an LSR carries the LDP Identifier for the label space the LSR intends to use for the interface and possibly additional information.

Receipt of an LDP Link Hello on an interface identifies a "Hello adjacency" with a potential LDP peer reachable at the link level on the interface as well as the label space the peer intends to use for the interface.

2.5.2. Extended Discovery Mechanism

LDP sessions between non-directly connected LSRs are supported by LDP Extended Discovery.

To engage in LDP Extended Discovery an LSR periodically sends LDP Targeted Hellos to a specific IP address. LDP Targeted Hellos are sent as UDP packets addressed to the well known LDP discovery port at the specific address.

An LDP Targeted Hello sent by an LSR carries the LDP Identifier for the label space the LSR intends to use and possibly additional optional information.

Extended Discovery differs from Basic Discovery in the following ways:

- A Targeted Hello is sent to a specific IP address rather than to the "all routers" group multicast address for the outgoing interface.

- Unlike Basic Discovery, which is symmetric, Extended Discovery is asymmetric.

  One LSR initiates Extended Discovery with another targeted LSR, and the targeted LSR decides whether to respond to or ignore the Targeted Hello. A targeted LSR that chooses to respond does so by periodically sending Targeted Hellos to the initiating LSR.

Receipt of an LDP Targeted Hello identifies a "Hello adjacency" with a potential LDP peer reachable at the network level and the label space the peer intends to use.

2.6. Establishing and Maintaining LDP Sessions

2.6.1. LDP Session Establishment

The exchange of LDP Discovery Hellos between two LSRs triggers LDP session establishment. Session establishment is a two step process:

- Transport connection establishment.
- Session initialization

The following describes establishment of an LDP session between LSRs LSR1 and LSR2 from LSR1's point of view. It assumes the exchange of Hellos specifying label space LSR1:a for LSR1 and label space LSR2:b for LSR2.

2.6.2. Transport Connection Establishment

The exchange of Hellos results in a Hello adjacency at LSR1 which binds the link (L) and the label spaces LSR1:a and LSR2:b.

1. If LSR1 does not already have an LDP session for the exchange of label spaces LSR1:a and LSR2:b it attempts to open an LDP TCP connection for a new session with LSR2.

   LSR1 determines the transport addresses to be used at its end (A1) and LSR2's end (A2) of the LDP TCP connection. Address A1 is determined as follows:

a) If LSR1 uses the Transport Address optional object to specify an address, A1 is the address LSR1 advertises via the optional object;

b) If LSR1 does not use the Transport Address optional object, A1 is the source IP address used for Hellos to LSR2.

Similarly, address A2 is determined as follows:

a) If LSR2 uses the Transport Address optional object (TLV), A2 is the address LSR2 advertises via the optional object;

b) If LSR2 does not use the Transport Address optional object, A2 is the source IP address used for Hellos from LSR2.

2. LSR1 determines whether it will play the active or passive role in session establishment by comparing addresses A1 and A2 as unsigned integers. If A1 > A2, LSR1 plays the active role; otherwise it is passive.

3. If LSR1 is active, it attempts to establish the LDP TCP connection by connecting to the well known LDP port at address A2. If LSR1 is passive, it waits for LSR2 to establish the LDP TCP connection to its well known LDP port.

2.6.3. Session Initialization

After LSR1 and LSR2 establish a transport connection they negotiate session parameters by exchanging LDP Initialization messages. The parameters negotiated include LDP protocol version, label distribution method, timer values, VPI/VCI ranges for label controlled ATM, DLCI ranges for label controlled Frame Relay, etc.

Successful negotiation completes establishment of an LDP session between LSR1 and LSR2 for the advertisement of label spaces LSR1:a and LSR2:b.

The following describes the session initialization from LSR1's point of view.

1. After the connection is established, if LSR1 is playing the active role, it initiates negotiation of session parameters by sending an Initialization message to LSR2. If LSR1 is

passive, it waits for LSR2 to initiate the parameter negotiation.

In general when there are multiple links between LSR1 and LSR2 and multiple label spaces to be advertised by each, the passive LSR cannot know which label space to advertise over a newly established TCP connection until it receives the first LDP PDU on the connection.

By waiting for the Initialization message from its peer the passive LSR can match the label space to be advertised by the peer (as determined from the LDP Identifier in the common header for the Initialization message) with a Hello adjacency previously created when Hellos were exchanged.

2. When LSR1 plays the passive role:

   a) If LSR1 receives an Initialization message it attempts to match the LDP Identifier carried by the message PDU with a Hello adjacency.

   b) If there is a matching Hello adjacency, the adjacency specifies the local label space for the session.

      Next LSR1 checks whether the session parameters proposed in the message are acceptable. If they are, LSR1 replies with an Initialization message of its own to propose the parameters it wishes to use and a KeepAlive message to signal acceptance of LSR2's parameters. If the parameters are not acceptable, LSR1 responds by sending a Nak message and closing the TCP connection.

   c) If LSR1 cannot find a matching Hello adjacency it sends a Nak message and closes the TCP connection.

   d) If LSR1 receives a KeepAlive in response to its Initialization message, the session is operational from LSR1's point of view.

   e) If LSR1 receives a Nak message, LSR2 has rejected its proposed session parameters and LSR1 closes the TCP connection.

3. When LSR1 plays the active role:

   a) If LSR1 receives a Nak message, LSR2 has rejected its proposed session parameters and LSR1 closes the TCP connection.

b) If LSR1 receives an Initialization message, it checks whether the session parameters are acceptable. If so, it replies with a KeepAlive message. If the session parameters are unacceptable, LSR1 sends a Nak message and closes the connection.

c) If LSR1 receives a KeepAlive message, LSR2 has accepted its proposed session parameters.

d) When LSR1 has received both an acceptable Initialization message and a KeepAlive message the session is operational from LSR1's point of view.

It is possible for a pair of incompatibly configured LSRs that disagree on session parameters to engage in an endless sequence of messages as each Naks the other's Initialization messages. An LSR must throttle its session setup retry attempts with an exponential backoff in situations where Initialization messages are being Nak'd. It is also recommended that an LSR detecting such a situation take action to notify an operator.

2.6.4. Initialization State Machine

It is convenient to describe LDP session negotiation behavior in terms of a state machine. We define the LDP state machine to have five possible states and present the behavior as a state transition table and as a state transition diagram.

Session Initialization State Transition Table

STATE	EVENT	NEW STATE
NON EXISTENT	Session TCP connection established established	INITIALIZED
INITIALIZED	Transmit Initialization msg	OPENSENT
	Receive acceptable Initialization msg Action: Transmit Initialization msg and KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Nak msg and close transport connection	NON EXISTENT
OPENREC	Receive KeepAlive msg	OPERATIONAL
	Receive Any other LDP msg Action: Transmit Nak msg and close transport connection	NON EXISTENT
OPENSENT	Receive acceptable Initialization msg Action: Transmit KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Nak msg and close transport connection	NON EXISTENT
OPERATIONAL	Receive Shutdown msg Action: Transmit Shutdown msg and close transport connection	NON EXISTENT
	Receive other LDP msgs	OPERATIONAL
	Timeout Action: Transmit Shutdown msg and close transport connection	NON EXISTENT

Session Initialization State Transition Diagram

```
                                    +------------+
                                    |            |
                      +------------>|NON EXISTENT|<--------------------+
                      |             |            |                     |
                      |             +------------+                     |
                      | Session        |    ^                          |
                      |   connection   |    |                          |
                      |   established  |    | Rx any LDP msg except    |
                      |                V    |   Init msg or Timeout    |
                      |            +-----------+                       |
         Rx Any other |            |           |                       |
            msg or    |            |INITIALIZED|                       |
            Timeout / |        +---|           |-+                     |
         Tx Nak msg   |        |   +-----------+ |                     |
                      |        | (Passive Role)  | (Active Role)       |
                      |        | Rx Init msg /   | Tx Init msg         |
                      |        | Tx Init msg     |                     |
                      |        |    Tx KeepAlive |                     |
                      |        V    msg          V                     |
                      |   +-------+        +--------+                  |
                      |   |       |        |        |                  |
                      +---|OPENREC|        |OPENSENT|----------------->|
                      +---|       |        |        | Rx Any other msg |
                      |   +-------+        +--------+    or Timeout    |
         Rx KeepAlive |        ^                |     Tx Nak msg       |
            msg       |        |                |                      |
                      |        |                | Rx Init msg /        |
                      |        +----------------+ Tx KeepAlive msg     |
                      |                                                |
                      |      +-----------+                             |
                      +----->|           |                             |
                             |OPERATIONAL|                             |
                             |           |---------------------------->+
                             +-----------+     Rx Shutdown msg
                      All other  |   ^            or TIMEOUT /
                        LDP msgs |   |         Tx Shutdown msg
                                 |   |
                                 +---+
```

2.6.5. Maintaining Hello Adjacencies

An LDP session with a peer has one or more Hello adjacencies.

An LDP session has multiple Hello adjacencies when a pair of LSRs are connected by multiple links that share the same label space; for example, multiple PPP links between a pair of routers. In this situation the Hellos an LSR sends on each such link carries the same LDP Identifier.

LDP includes mechanisms to monitor the necessity of an LDP session and its Hello adjacencies.

LDP uses the regular receipt of LDP Discovery Hellos to indicate a peer's intent to use the label space identified by the Hello. An LSR maintains a hold timer with each Hello adjacency which it restarts when it receives a Hello that matches the adjacency. If the timer expires without receipt of a matching Hello from the peer, LDP concludes that the peer no longer wishes to label switch using that label space for the link (or target, in the case of Targeted Hellos) in question or that the peer has failed, and it deletes the Hello adjacency. When the last Hello adjacency for a LDP session is deleted, the LSR terminates the LDP session by closing the transport connection.

2.6.6. Maintaining LDP Sessions

LDP includes mechanisms to monitor the integrity of the session transport connection.

LDP uses the regular receipt of LDP PDUs on the session transport connection to monitor the integrity of the connection. An LSR maintains a keepalive timer for each peer session which it resets whenever it receives an LDP PDU from the session peer. If the keepalive timer expires without receipt of an LDP PDU from the peer the LSR concludes that the transport connection is bad or that the peer has failed, and it terminates the peer session by closing the transport connection.

An LSR must arrange that its LDP peer sees an LDP PDU from it at least every keepalive time period to ensure the peer restarts the session keepalive timer. The LSR may send any protocol message to meet this requirement. In circumstances where an LSR has no other information to communicate to its peer, it sends a KeepAlive message.

An LSR may choose to terminate an LDP session with a peer at any time. Should it choose to do so, it informs the peer with a Shutdown

message.

2.7. Label Distribution and Management

2.7.1. Label Distribution Control Mode

The behavior of the initial setup of LSPs is determined by whether the LSR is operating with independent or ordered LSP control. An LSR may support both types of control as a configurable option.

2.7.1.1. Independent Label Distribution Control

When using independent LSP control, each node may advertise label mappings to its neighbors at any time it desires. For example, when operating in independent Downstream-on-Demand mode, an LSR may answer requests for label mappings immediately, without waiting for a label mapping from the next hop. When operating in independent Downstream allocation mode, an LSR may advertise a label mapping for a FEC to its neighbors whenever it is prepared to label-switch that FEC.

A consequence of using independent mode is that an upstream label can be advertised before a downstream label is received. This can result in unlabeled packets being sent to the downstream node.

2.7.1.2. Ordered Label Distribution Control

When using LSP ordered control, an LSR may initiate the transmission of a label mapping only for an FEC for which it has a label mapping for the FEC next hop, or for which the LSR is the egress. For each FEC for which the LSR is not the egress and no mapping exists, the LSR MUST wait until a label from a downstream LSR for is received before mapping the FEC and passing corresponding labels to upstream LSRs.

An LSR may be an egress for some FECs, and a non-egress for others. An LSR may act as an egress LSR, with respect to a particular FEC, under any of the following conditions:

1.   The FEC refers to the LSR itself (including one of its directly attached interfaces).

2.   The next hop router for the FEC is outside of the Label Switching Network.

3   FEC elements are reachable by crossing a routing domain boun-dary, such as another area for OSPF summary net-works, or another autonomous system for OSPF AS externals and BGP routes

[rfc1583] [rfc1771].

2.7.2. Label Retention Mode

2.7.2.1. Conservative Label Retention Mode

In Downstream Allocation mode, label mapping advertisements for all routes may be received from all peer LSRs. When using conservative label retention, advertised label mappings are only retained if they will be used to forward packets (i.e., if they are received from a valid next hop according to routing). If operating in Downstream-on-Demand mode, label mappings will only be requested of the appropriate next hop LSR according to routing. Since Downstream-on-Demand mode is primarily used when label conservation is desired (e.g., an ATM switch with limited cross connect space), it is typically used with the conservative label retention mode.

The main advantage of the conservative mode is that the only the labels that are required for the forwarding of data are allocated and maintained. This is particularly important in LSRs where the label space is inherently limited, such as in an ATM switch. A disadvantage of the conservative mode is that if routing changes the next hop for a given destination, a new label must be obtained from the new next hop before labeled packets can be forwarded.

2.7.2.2. Liberal Label Retention Mode

In Downstream Allocation mode, label mapping advertisements for all routes may be received from all peer LSRs. When using liberal label retention, advertised label mappings are retained from all next hops regardless of whether they are valid next hops for the advertised mapping. When operating in Downstream-on-Demand mode, label mappings are requested of all peer LSRs. Note, however, that Downstream-on-Demand mode is typically associated with ATM switch-based LSRs where the conservative approach is recommended.

The main advantage of the liberal label retention mode is that reaction to routing changes can be quick because labels already exist. The main disadvantage of the liberal mode is that unneeded label mappings are distributed and maintained.

2.7.3. Label Advertisement Mode

Each interface on an LSR is configured to operate in either Downstream or Downstream-on-Demand allocation mode. LSRs exchange advertisement modes during initialization. The major difference between Downstream and Downstream-on-Demand modes is in which LSR takes responsibility for initiating mapping requests and mapping advertisements

2.8. LDP Identifiers and Next Hop Addresses

An LSR maintains learned labels in a Label Information Base (LIB). When operating in Downstream (as opposed to Downstream-on-Demand) more, the LIB entry for an address prefix associates a collection of (LDP Identifier, label) pairs with the prefix, one such pair for each peer advertising a label for the prefix.

When the next hop for a prefix changes the LSR must retrieve the label advertised by the new next hop from the LIB for use in forwarding. To retrieve the label the LSR must be able to map the next hop address for the prefix to an LDP Identifier.

Similarly, when the LSR learns a label for a prefix from an LDP peer, it must be able to determine whether that peer is currently a next hop for the prefix to determine whether it needs to start using the newly learned label when forwarding packets that match the prefix. To make that decision the LSR must be able to map an LDP Identifier to the peer's addresses to check whether any are a next hop for the prefix.

To enable LSRs to map between a peer LDP identifier and the peer's addresses, LSRs advertise their addresses using LDP Address and Withdraw Address messages.

An LSR sends an Address message to advertise its addresses to a peer. An LSR sends a Withdraw Address message to withdraw previously advertised addresses from a peer

2.9. Loop Detection

Each LSR MUST support the configurable loop-detection option. LSRs perform loop detection via the LSR-path-vector object (TLV) contained within each Mapping and Query message. Upon receiving such a message, the LSR performs loop detection by verifying that its unique router-id is not already present in the list. If a loop is detected, the LSR must transmit a NAK message to the sending node, and does

not install the mapping or propagate the message any further. In addition, if there is an upstream label spliced to the downstream label for the FEC, the LSR must unsplice the labels. On those messages in which no loop is detected, the LSR must concatenate itself to the LSR-path-vector before propagating.

If loop detection is desired in some portion of the network, then it should be turned on in ALL LSRs within that portion of the network, else loop detection will not operate properly.

2.10. Loop Prevention via Diffusion

LSR diffusion support is a configurable option, which permits an LSR to verify that a new routed path is loop free before installing an LSP on that path. An LSR which supports diffusion does not splice an upstream label to a new downstream label until it ensures that concatenation of the upstream path with the new downstream path will be loop free.

A LSR which detects a new next hop for an FEC transmits a Query message containing its unique router id to each of its upstream peers. An LSR that receives such a Query message processes the Query as follows. (The following procedures are described in terms of Ack and Nak messages. An Ack is a Notification message signalling Success; a Nak is a Notification message signalling Loop Detected)

- o If the downstream LSR not the correct next hop for the given FEC, the upstream LSR responds with an Ack message, indicating that the downstream LSR may change to the new path.

- o If the downstream LSR is the correct next hop for the given FEC, the upstream LSR performs loop detection via the LSR-path-vector.

- o If a loop is detected, the upstream LSR responds with a Nak message that indicates the LSR is to be "pruned, and the LSR unsplices all connections for that FEC to the downstream node, thereby pruning itself off of the tree.

- o If a loop is not detected, the upstream node concatenates its unique router-id to the LSR-path-vector, and propagates the Query message to its upstream peers.

- o Each LSR which receives an Ack message from its upstream peer in response to a query message, in turn forwards the acknowledgement to the downstream LSR which sent the Query message.

- If an LSR doesn't receive a Ack Message for a given query within a "reasonable" period of time, it "unsplices" the upstream peer that has not responded, and responds with a Nak message to its downstream peer, indicating the pruning of the upstream peer.

- An LSR which receives a new Query message for an FEC before it has received responses from all of its upstream peers for a previous Query message must concatenate the old and the new LSR-path-vector within the new query advertisement before propagating.

- The diffusion computation continues until each upstream path responds with an acknowledgment. An LSR that does not have any upstream LDP peers must acknowledge the Query message.

The LSR which began the diffusion may splice its upstream label to the new downstream label only after receiving an acknowledge message from the upstream peer.

As LSR diffusion support is a configurable option, an LSR which does not support diffusion will never originate a Query message. However, these LSRs must still recognize and process the Query messages, as described above.

2.11. Explicitly Routing LSPs

The need for explicit routing (ER) in MPLS has been explored elsewhere [ARCH] [FRAME]. At the MPLS WG meeting held during the Washington IETF there was consensus that LDP should support explicit routing of LSPs with provision for indication of associated (forwarding) priority. This section specifies mechanisms to provide that support, and provides a means to allow the reservation of 'resources' for the explicitly routed LSP.

In this document we propose an end to end setup mechanism that could, in principal, be invoked from either end of the explicitly routed LSP (ERLSP). However we specify it here only for the case of initiation by the ingress in the belief that such a mechanism maps naturally to the setup in the opposite direction. We believe that the, inevitable, latency associated with this (end to end) setup mechanism is tolerable since most of the motivations for ERLSPs, for example 'traffic engineering' imply that the LSPs setup in this manner will have a long lifetime (at least when compared to those setup in response to dynamic routing).

We introduce objects and procedures that provide support for:

- Strict and Loose explicit routing

- Specification of class of service

- Reservation of bandwidth

- Route pinning

- ERLSP preemption

Only unidirectional point-to-point ERLSP is specified currently. The scheme can be easily extended to accommodate multipoint-to-point ERLSPs. The FEC object (TLV) may be used to determined which ERLSPs are "merged" to form a multipoint-to- point ERLSP. Alternatively, a multipoint-to-point ERLSP can be setup from the egress by completely specifying the multipoint- to-point tree. Also, tunneling ERLSPs within other ERLSPs is for future study.

To setup a ERLSP an LSR (that will be the 'ingress' of the LSP) generates an explicit request. The explicit request contains an explicit route object which in turn contains a sequence of explicit request next hop objects and a pointer to the current entry in that sequence. The explicit request next hop objects specify the IP address of the LSRs through which the ERLSP should pass. These LSR hops specified in the explicit route are referred to as 'peg LSRs'.

An explicit request MUST specify the stream that will be associated with the ERLSP by inserting the appropriate FEC value in the request. The FEC value 'opaque tunnel' exists to support ERLSPs where the intermediate LSRs on the LSP need know nothing about the traffic flowing on the LSP.

The setup mechanism for ERLSPs employs an end to end protocol. Individual ERLSPs are uniquely identified by an ERLSPID associated with them by the LSR that initiates their setup. The ERLSPID is generated by the ingress LSR of the LSP. The ERLSPID has another component called Peg ERLSPID which is generated by each peg LSR when the next peg LSR from itself is loosely routed. This is used by the intermediate LSRs to identify a loosely routed segment. The Peg ERLSPID is not used in a segment that is strictly routed. Requests travel from the 'ingress' of the LSP toward what will be the 'egress'. Responses indicating the status of the ERLSP request travel back toward the ingress of the ERLSP. ERLSPID is used in both Request and Response messages.

The addresses specified in the next hop objects in the explicit

route object should be those of the LSR's IP address or the incoming interfaces on the LSRs through which the LSP should pass. The ERLSPID, FEC, incoming interface (previous hop) and LDP identifier of the LSR that generated the message are all stored in an ERLSP control block. Here's a synopsis of the entire mechanism to instantiate an ERLSP:

An ingress node originates a ERLSP request message. The message contains an unique ERLSPID, FEC object, explicit route object, and an optional object for resource assignment for the ERLSP.

At an intermediate node the 'active' ERNH object is identified by the pointer in the explicit route object. On message receipt the pointer always points to the receiving LSR object in the explicit route message in case of strict routing. If a segment of ERLSP is loosely routed then pointer always points to the upstream peg LSR at all the intermediate LSRs in this segment. The penultimate hop to the downstream peg LSR advances the pointer to the next ERNH object in the list.

If the ERNH objects subtype indicates 'Strict' then dependent on the next ERNH IP address the appropriate LDP Identifier for the LDP session with the next hop and the appropriate output interface are discovered (by using the information learnt from the address message see Section "LDP Identifiers"). The outgoing interface (next hop) information is also stored in the ERLSP control block. In the case of strict ERLSP, the neighbor MUST be directly adjacent to the current LSR.

If the ERNH object subtype indicates 'Loose' then dependent upon the next ERNH IP address a next hop is selected as per the FIB information for the downstream peg LSR. This information is again maintained in the ERLSP control block. Peg LSRs are allowed to change the Explicit Route Object if the path to the next Peg LSR is selected to be 'loose'. This allows the Peg LSRs to select a specific path to the next Peg LSR. The default path to the next Peg LSR in case the segment is chosen as 'loose' is determined by the hop-by- hop forwarding path to the next Peg LSR. However, Peg LSR are allowed only to select a path downstream to the next Peg LSR, they cannot change paths on any other segment of the ERLSP.

Bandwidth reservations (if any) are processed. How this happens, i.e. the precise connection admission procedures is outside the scope of the LDP specification. The admission control must also use the preemption value specified for the LSP in determining if resources are available for the LSP. If a reservation cannot be accommodated a response indicating that fact is

returned to the previous hop. Note that the resources are only reserved at this time. The LSRs will commit the bandwidth with the labels when the response comes back from the egress LSR.

If the ERLSP can be accommodated the pointer in the explicit request object is incremented to point at the next explicit request next hop object in case of strict routing and the request message is sent to the LDP peer discovered as described above. In case of loose routing, the pointer is incremented only if the direct next hop is the next downstream peg LSR.

If an LSR finds it impossible to satisfy a Explicit request then an 'Explicit response' message is created indicating the reason. The ERLSPID from (failed) request is inserted in the message and it is sent to the LDP peer identified in the associated entry in the ERLSP control block after which the ERLSP block is freed.

LSRs receiving Explicit responses indicating failure process them in a similar manner. They create a new Explicit request and copy the ERLSPID and Status from the Explicit request they received into it. They use the ERLSPID to obtain the appropri- ate ERLSP control block and thus identify the LDP peer toward which the 'new' Explicit response message should be sent. Hav- ing done that they free the ERLSP control block.

When an Explicit request reaches the LSR specified in the last ERNH object in that request and that LSR accedes to the request it generates an Explicit response indicating successful setup of the ERLSP. The egress node also includes a label in the response message. The Explicit response is (reverse path) for- warded through the LSRs that the original Explicit request traversed using the mechanism described above (inspection of ERLSP control block). In this case, of course, the ERLSP con- trol block is not deleted. An intermediate LSR receiving such a response message allocates a new label on its incoming interface and creates a connection between the new and the given label in the message. The LSR also commits the previously reserved bandwidth to this connection at the appropriate scheduler(s). The LSR then forwards the message to its previous hop with the new label. When the successful response reaches the ingress LSR the ERLSP is declared in-service.

There is also support for route pinning for loosely routed seg- ments. When a ERLSP is pinned the loose path is not changed when `better' paths become available. Once a ERLSP goes in- service there is protocol support to reassign resources to the ERLSP if required.

2.12. ERLSP State Machine

The ERLSP control block may contain the following information:

- ERLSPID/Peg ERLSPID
- State
- FEC object
- Flags
  - o Self is Peg Node
  - o Pinned path
  - o Upstream segment (Strict/Loose) type
  - o downstream segment (Strict/Loose) type
- next peg node
- preemption level
- upstream neighbor (next hop/interface)
- downstream neighbor (next hop/interface)
- BW information (only at peg LSRs with loose downstream segment)
- Explicit Route Object (only at peg LSRs with loose downstream segment)

For the purpose of matching message to existing ERLSP control block, both the ERLSPID and Peg ERLSPID in the message are matched against the ones in the control block. Its only when both of them match that the message is considered to be for the matched control block, otherwise it is treated as a new ERLSP request. The ingress may use the ERLSPID as the peg ERLSPID. At the peg nodes, the control block fields ERLSPID and Previous Peg ERLSDID are compared because Peg ERLSPID contains the self assigned Peg ERLSPID. Also note that the Request message at Peg node is only compared for ERLSPID to select a control block.

The state tables for peg node and non peg nodes are given separately. Separate state tables are used only for illustrative purposes. The state engines can be collapsed into a single state engine. Moreover, a completely strict ERLSP can be treated as a special case of loosely routed where every neighbor is a peg LSR with several of the state transitions optimized.

2.12.1. Loose Segment Peg LSR Transitions:

Peg LSRs in a loosely routed ERLSP segment are those that are explicitly listed in the explicit route object as the starting or ending of a loose segment.

State NULL

Event	Action	New State
Request	Create ERLSP control block; store relevant information from the message into the control block; select a new peg ERLSPID; reserve BW specified in the message; obtain next hop (or interface) towards next peg LSR; propagate message towards the obtained next hop.	Response Awaited
	If last node in the explicit route object, allocate an upstream label; commit BW; originate a Response message upstream.	Established
	If unable to process request for any reason, issue a NAK message to the sender with appropriate error code.	No change
Response	Send NAK message to the sender.	No change
Others	Silently ignore event.	No change

State RESPONSE_AWAITED

Event	Action	New State
Response	Install downstream label in message; choose an upstream label; connect upstream to downstream label; commit BW to the connection; propagate Response upstream with upstream label.	Established
	If unable to process Response message for any reason then recover resources; originate a Nak message	Null

	upstream; originate a Release message downstream; delete control block.	
Upstream lost	Release resources; propagate Nak downstream; delete control block.	Null
Downstream lost	Reassign a new Peg ERLSPID. Start RETRY timer.	Retry
Nak from downstream	Reassign a new Peg ERLSPID. RETRY timer.	Retry
	If error code in Nak is severe then propagate the Nak upstream; release resources; delete control block.	Null
Nak from upstream	Release resources; propagate Nak downstream; delete control block.	Null
New NH	If ERLSP is pinned, ignore event. Otherwise, send a Nak downstream; change NH in the control block; reassign a new Peg ERLSPID. Start RETRY timer.	Retry
Others	Silently ignore event.	No change

State RETRY

Event	Action	New State
Retry Timer	Originate Request message towards the next hop in the control block.	Response Awaited
New NH	If ERLSP is pinned, ignore the event. Otherwise change next hop information in the control block.	No change
Nak from upstream	Release all resources (BW, label, timer); delete control block.	Null
Upstream lost	Release all resources (BW, label, timer); delete control block.	Null
Release	Release all resources (BW, label, timer); delete control block.	Null

Downstream lost	If there is a new next hop, update that in the control block.	No change
	Otherwise, delete timer; recover resources; send Nak upstream; delete control block.	Null
Others	Silently ignore event.	No change

State RECONNECT_AWAITED

Event	Action	New State
Request	Make appropriate changes in the control block; make label connection; send a Response message upstream with upstream label.	Established
	If unable to process Request message for any reason then send a Release message downstream and a Nak message upstream; release resources; delete control block.	Null
Reconnect Awaited Timer	Release resources; send Release message downstream; delete control block.	Null
Upstream lost	Ignore event.	No change
Downstream lost	Release resources; delete control block.	Null
New NH	Release resources; delete control block.	Null
Nak from downstream	Release resources; delete control block.	Null
Others	Silently ignore event.	No change

State ESTABLISHED

Event	Action	New State

Upstream lost	Start RECONNECT_AWAITED timer.	Reconnect Awaited
Downstream lost	Reassign a new Peg ERLSPID. Start RETRY timer.	Retry
Nak from downstream	Reassign a new Peg ERLSPID. Start RETRY timer.	Retry
	If error code in Nak is severe then propagate the Nak upstream; release resources; delete control block.	Null
Nak from upstream	Reassign a new Peg ERLSPID. Start RECONNECT_AWAITED timer.	Reconnect Awaited
	If error code in Nak is severe, then propagate the Nak downstream; release resources; delete control block.	Null
New NH	If ERLSP is pinned, ignore the event. Otherwise, send a Nak downstream; change next hop in control block; reassign a new Peg ERLSPID. Start RETRY timer.	Retry
Release	Release resources; propagate message downstream; delete control block.	Null
Others	Silently ignore event.	No change

2.12.2. Loose Segment Non-Peg LSR Transitions:

Non-peg LSRs in a loose segment of an ERLSP are the LSRs intermediate to two peg LSRs and through which the loose segment is routed using the hop-by-hop forwarding path.

State NULL

Event	Action	New State
Request	Create ERLSP control block; reserve BW specified in the message; obtain next hop (or interface) towards next peg LSR; if penultimate hop to next peg LSR then increment pointer in ERNH object; propagate message towards the obtained next hop	Response Awaited
	If unable to process request for any reason, issue a Nak message to the sender with appropriate error code.	No change
Response	Send a Nak message to the sender.	No change
Others	Silently ignore event.	No change

State RESPONSE_AWAITED

Event	Action	New State
Response	Install downstream label in message; choose an upstream label; connect upstream to downstream label; commit BW to connection; propagate Response upstream with upstream label.	Established
	If unable to process Response message for any reason then recovery resources; propagate a Nak message upstream; originate a Release message downstream; delete control block.	Null
Upstream lost	Originate a Nak message downstream; delete control block.	Null

Downstream lost	Originate a Nak message upstream; delete control block.	Null
Nak from downstream	Propagate Nak message upstream; release reserved BW; delete control block.	Null
Nak from upstream	Propagate Nak message downstream; release reserved BW; delete control block;	Null
New NH	If ERLSP is pinned, ignore the event. Otherwise, send Nak message upstream and downstream; release reserved BW; delete control block.	Null
Release	Propagate message downstream; release resources; delete control block.	Null
Others	Silently ignore event.	No change

State ESTABLISHED

Event	Action	New State
Upstream lost	Send Nak message downstream; release resources (BW, label); delete control block.	Null
Downstream lost	Send Nak message upstream; release resources; delete control block.	Null
Nak from downstream	Release resources; propagate Nak message upstream; delete control block.	Null
Nak from upstream	Release resources; propagate message Nak downstream; delete control block.	Null
New NH	If ERLSP is pinned, ignore the event. Otherwise, release resources; originate Nak message upstream; originate Nak message downstream; delete control block.	Null

Release	Release resources; propagate message downstream; delete control block.	Null
Others	Silently ignore event.	No change

2.12.2.1. Strict Segment Transitions

A LSR whose upstream and downstream segment of an ERLSP is strict has a state transition exactly similar to the non-peg LSR (only different being does not handle the case of pinned down option).

2.12.3. ERLSP Timeouts

The following timeouts are used in the state transition:

RETRY
Default value TBD. This timer is set by the peg LSR to originate a Request message downstream on the elapse of the timer when a downstream loose segment is lost.

RECONNECT
Default value TBD. This timer is set by the peg LSR to deinstall an ERLSP on the elapse of the timer when a upstream loose segment is lost.

2.12.4. ERLSP Error Codes

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

To be supplied.

This subsection should be moved to Section 3.

END NOTE * END NOTE * END NOTE:

3. Protocol Specification

Previous sections that describe LDP operation have discussed scenarios that involve the exchange of messages among LDP peers. This section specifies the message encodings and procedures for processing the messages.

LDP message exchanges are accomplished by sending LDP protocol data units (PDUs) over LDP session TCP connections.

Each LDP PDU can carry one or more LDP messages. Note that the messages in an LDP PDU need not be related to one another. For example, a single PDU could carry a message advertising FEC-label bindings for several FECs, another message requesting label bindings for several other FECs, and a third notification message signalling some event.

3.1. LDP PDUs

Each LDP PDU is a fixed LDP header followed by one or more LDP messages. The fixed LDP header is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Version                      |         PDU Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         LDP Identifier                        |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |          Res                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version
  Two octet unsigned integer containing the version number of the protocol. This version of the specification specifies LDP protocol version 1.

PDU Length
  Two octet integer specifying the total length of this PDU in bytes, excluding the Version and PDU Length fields.

LDP Identifier
  Six octet field that uniquely identifies the label space for which this PDU applies. The first four octets encode an IP address assigned to the LSR. This address should be the router-id, also used in LSR Path Vector used by loop detection and loop prevention

procedures. The last two octets identify a label space within the LSR. For a platform-wide label space, these should both be zero.

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

3.2. Type-Length-Value Encoding

LDP uses a Type-Length-Value (TLV) encoding scheme to encode much of LDP message contents. An LDP TLV is encoded as a 2 octet Type field, followed by a 2 octet Length Field followed by a variable length Value field.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type                      |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          Value                                |
~                                                               ~
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type
Encodes how the Value field is to be interpreted.

Length
Specifies the length of the Value field in octets.

Value
Octet string of Length octets that encodes information the interpretation of which is specfied by the Type field.

Note that the Value field itself may contain TLV encodings. That is, TLVs may be nested.

The TLV encoding scheme is very general. In principle, everything appearing in an LDP PDU could be encoded as a TLV. This specifica- tion does not use the TLV scheme to its full generality. It is not used where its generality is unnecessary and its use would waste space unnecessarily. These are usually places where the type of a

value to be encoded is known, for example by its position in a message or an enclosing TLV, and the length of the value is fixed or readily derivable from the value encoding itself.

Some of the TLVs defined for LDP are similar to one another. For example, there is a Generic Label TLV, an ATM Label TLV, and a Frame Relay TLV; see Sections "Generic Label TLV", "ATM Label TLV", and "Frame Relay TLV".

While is possible to think about TLVs related in this way in terms of a TLV type that specifies a TLV class and a TLV subtype that specifies a particular kind of TLV within that class, this specification does not formalize the notion of a TLV subtype.

The specification assigns type values for related TLVs, such as the label TLVs, from of a contiguous block in the 16-bit TLV type number space.

Section "TLV Summary" lists the TLVs defined in this version of the protocol and the document section that describes each.

3.3. Commonly Used TLVs

There are several TLV encodings used by more than one LDP message. The encodings for these commonly used TLVs are specified in this section.

3.3.1. FEC TLV

Labels are bound to Forwarding Equivalence Classes (FECs). An FEC is a list of one or more FEC elements. The FEC TLV encodes FEC items.

Its encoding is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     FEC (0x0100)              |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        FEC Element 1                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        FEC Element n                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

FEC Element 1 to FEC Element n

There are several types of FEC elements; see Section "FEC Types". The FEC element encoding depends on the type of FEC element. Note that while the representation of the FEC element value is type-dependent that the value encoding itself is one where standard LDP TLV encoding is not used.

A FEC Element value is encoded as a 1 octet field that specifies the element type, and a variable length field that is the type-dependent element value.

The FEC Element value encoding is:

FEC Element type name	Type	Value
Wildcard	0x01	No value; i.e., 0 value octets; see below.
Prefix	0x02	See Prefix value encoding below.
Router Id	0x03	4 octet full IP address.
Flow	0x04	See Flow value encoding below.

Wildcard FEC Element

To be used only in the Label Withdraw and Label Release Messages. Indicates the withdraw/release is to be applied to all FECs associated with the label within the following label TLV. Must be the only FEC Element in the FEC TLV.

Prefix FEC Element value encoding:

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     Address Family            |    PreLen     |               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
     |                                                               |
     |                            Prefix                             |
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Family
  Two octet quantity containing a value from ADDRESS FAMILY
  NUMBERS in Assigned Numbers [ref] that encodes the address fam-
  ily for the address prefix in the Prefix field.

PreLen
  One octet unsigned integer containing the length in bits of the
  address prefix that follows.

Prefix
  An address prefix encoded according to the Address Family
  field, whose length, in bits, was specified in the PreLen
  field, padded to a byte boundary.

Flow FEC Element value encoding:

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     Network Source Address                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     Network Destination Address               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |         Source Port           |         Dest Port             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    Protocol   |   Direction   |         Reserved              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Network Source Address
  Four octet source IPv4 address.

Network Destination Address
  Four octet destination IPv4 address.

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

For generality the address encodings here should include an
Address Family field, etc.

END NOTE * END NOTE * END NOTE:

Source Port
Two octet source port.

Destination Port
Two octet destination port.

Protocol
Protocol type.

Direction
One octet indicating the direction of the LSP. Field is set to
1 on Downstream; field is set to 2 on Upstream.

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

Use of this FEC is not fully specified in this version of the
protocol

END NOTE * END NOTE * END NOTE:

3.3.1.1. FEC Procedures

If in decoding a FEC TLV an LSR encounters a FEC Element type it can-
not decode, it should stop decoding the FEC TLV, abort processing the
message containing the TLV, and send an Ack/Nack message to its LSR
peer signalling an error.

3.3.2. Label TLVs

Label TLVs encode labels. Label TLVs are carried by the messages
used to advertise, request, release and withdraw label mappings.

There are several different kinds of Label TLVs which can appear in
situations that require a Label TLV.

3.3.2.1. Generic Label TLV

An LSR uses Generic Label TLVs to encode labels for use on links for which label values are independent of the underlying link technology. Examples of such links are PPP and Ethernet.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Generic Label (0x0200)    |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Label                                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Label
  This is a 20-bit label value as specified in [ENCAP] represented as a 20-bit number in a 4 octet field.

3.3.2.2. ATM Label TLV

An LSR uses ATM Label TLVs to encode labels for use on ATM links.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ATM Label (0x0201)        |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res| V |          VPI          |              VCI              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

V-bits
  Two-bit switching indicator. If V-bits is 00, both the VPI and VCI are significant. If V-bits is 01, only the VPI field is signifi- cant. If V-bit is 10, only the VCI is significant.

VPI
  Virtual Path Identifier. If VPI is less than 12-bits it should be right justified in this field and preceding bits should be set to 0.

VCI
  Virtual Connection Identifier. If the VCI is less than 16- bits, it should be right justified in the field and the preceding bits must be set to 0. If Virtual Path switching is indicated in the V-bits field, then this field must be ignored by the receiver and set to 0 by the sender.

3.3.2.3. Frame Relay Label TLV

An LSR uses Frame Relay Label TLVs to encode labels for use on Frame Relay links.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Frame Relay Label (0x0202)  |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved       |Len|                    DLCI                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Len
  This field specifies the number of bits of the DLCI. The following values are supported:
     0 = 10 bits DLCI
     1 = 17 bits DLCI
     2 = 23 bits DLCI

DLCI
  The Data Link Connection Identifier. Refer to draft-ietf-mpls-fr-01.txt [FR] for the label values and formats.

3.3.3. Address List TLV

The Address List TLV appears in Address and Address Withdraw messages.

Its encoding is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address List (0x0101)     |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address Family            |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   |                        Addresses                              |
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Family
  Two octet quantity containing a value from ADDRESS FAMILY NUMBERS in Assigned Numbers [ref] that encodes the addresses contained in the Addresses field.

Addresses
  A list of addresses from the specified Address Family. The encoding of the individual addresses depends on the Address Family.

  The following address encodings are defined by this version of the protocol:

Address Family	Address Encoding
IPv4	4 octet full IPv4 address

3.3.4. COS TLV

The COS (Class of Service) TLV may appear as an optional field in messages that carry label mappings. Its encoding is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     COS (0x0102)              |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |     COS Value                                                 |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

COS Value
  The COS Value may be one of several types, encoded as a 1 octet
  type followed by a variable length, type-dependent value. Note
  that the encoding of the COS value is not the standard LDP TLV
  encoding. Note also that the length of the type-dependent value
  can be derived from the length of the COS TLV.

  The following COS value encodings are defined by this version of
  the protocol:

COS Name	Type code	Value
IP Prec	0x01	1 octet IP Precedence

If in decoding a COS TLV an LSR encounters a COS type it cannot
decode, it should stop decoding the COS TLV, abort processing the
message containing the TLV, and send an Ack/Nack message to its LSR
peer signalling an error.

3.3.5. Hop Count TLV

The Hop Count TLV appears as an optional field in messages that set
up LSPs. It calculates the number of LSR hops along an LSP as the
LSP is being setup.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Hop Count (0x0103)        |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     HC Value  |
+-+-+-+-+-+-+-+-+
```

HC Value
  1 octet unsigned integer hop count value.

3.3.5.1. Hop Count Procedures

During setup of an LSP an LSR may receive a Label Mapping or Label
Request message for the LSP that contains the Hop Count TLV. If it
does, it should record the hop count value. If the LSR then passes a
Label Mapping message for the LSP to an upstream peer or a Label

Request to a downstream peer to continue the LSP setup, it must increment the recorded hop count value and include it in a Hop Count TLV in the message. The first LSR in the LSP should set the hop count value to 1.

If an LSR receives a Label Mapping message containing a Hop Count TLV, it must check the hop count value to determine whether the hop count has wrapped (hop count value = 0). If so, it must reject the Label Mapping message in order to prevent a forwarding loop.

3.3.6. Path Vector TLV

The Path Vector TLV is used in messages that implement LDP loop detection and prevention. It records the path of LSRs a label advertisement has traversed to setup an LSP. Its encoding is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Path Vector (0x0104)        |      Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            LSR Id 1                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            LSR Id n                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

One or more LSR Ids
  A list of router-identifiers indicating the path of LSRs the mapping message has traversed. Each router-id must be the router-id component of the LDP identifier for the corresponding LSR. This ensures it is unique within the LSR network.

3.3.6.1. Path Vector Procedures

During setup of an LSP an LSR may receive a Label Mapping message for the LSP that contains the Path Vector TLV. If it does, the LSR must pass a Label Mapping message for the LSP to the upstream peer(s) to continue the LSP setup. This message must include a Path Vector TLV in the message. The value of the path vector in the Path Vector TLV

must be the received path vector with the LSRs own LSR Id appended to it.

If an LSR receives a Label Mapping message containing a Path Vector TLV, it must check the path vector value to determine whether the vector contains its own LSR-id. If so, it must reject the Label Mapping message in order to prevent a forwarding loop.

The Path Vector TLV is also used in the Label Query message. See Sections "Loop Detection" and "Loop Prevention via Diffusion" for more details.

3.3.7. Status TLV

Notification messages carry Status TLVs to specify events being signalled.

The encoding for the Status TLV is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Status (0x0300)           |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Status Code                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Message Type             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Status Code
  32-bit unsigned integer encoding the event being signalled. The structure of a Status Code is:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |F|E|                 Status Data                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

  F bit
    Fatal error bit. If set (=1), this is a fatal error notification. If clear (=0), this is an advisory notification.

E bit
End-to-end bit. If set (=1), the notification should be forwarded to the LSR for the next-hop or previous-hop for the LSP, if any, associated with the event being signalled. If clear (=0), the notification should not be forwarded.

Status Data
30-bit unsigned integer which specifies the status information.

This specification defines Status Codes (32-bit unsigned integers with the above encoding).

A Status Code of 0 signals success.

Message ID
If non-zero, 32-bit value that identifies the peer message to which the Status TLV refers. If zero, no specific peer message is being identified.

Message Type
If non-zero, the type of the peer message to which the Status TLV refers. If zero, the Status TLV does not refer to any specific peer message.

3.4. LDP Messages

All LDP messages have the following TLV format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Message Type              |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                     Mandatory Parameters                      |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                     Optional Parameters                       |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type
  Identifies the type of message

Message Length
  Specifies the length of the message value component (Mandatory plus Optional Parameters) in octets

Message Id
  Four octet integer used to identify this message. Used by the sending LSR to facilitate identifying notification messages that may apply to this message. An LSR sending a notification message in response to this message will include this Message Id in the notification message; see Section "Notification Message".

Mandatory Parameters
  Variable length set of required message parameters. Some messages have no required parameters.

  For messages that have required parameters, the required parameters MUST appear in the order specified by the individual message specifications in the sections that follow.

Optional Parameters
  Variable length set of optional message parameters. Many messages have no optional parameters.

  For messages that have optional parameters, the optional parameters may appear in any order.

The following message types are defined in this version of LDP:

Message Name	Type	Section Title
Notification	0x0001	"Notification Message"
Hello	0x0100	"Hello Message"
Initialization	0x0200	"Initialization Message"
KeepAlive	0x0201	"KeepAlive Message"
Address	0x0300	"Address Message"
Address Withdraw	0x0301	"Address Withdraw Message"
Label Mapping	0x0401	"Label Mapping Message"
Label Request	0x0402	"Label Request Message"
Label Withdraw	0x0403	"Label Withdraw Message"
Label Release	0x0404	"Label Release Message"
Label Query	0x0405	"Label Query Message"
Explicit Route Request	0x0500	"Explicit Route Request Message"
Explicit Route Response	0x0501	"Explicit Route Response Message"

The sections that follow specify the encodings and procedures for these messages.

Some of the above message are related to one another, for example the Label Mapping, Label Request, Label Withdraw, and Label Release messages.

While is possible to think about messages related in this way in terms of a message type that specifies a message class and a message subtype that specifies a particular kind of message within that class, this specification does not formalize the notion of a message subtype.

The specification assigns type values for related messages, such as the label messages, from of a contiguous block in the 16-bit message type number space.

3.4.1. Notification Message

An LSR sends a Notification message to inform an LDP peer of a significant event. A Notification message signals a fatal error or provides advisory information regarding an item such as the processing of LDP messages or the state of the LDP session.

The encoding for the Notification Message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Notification (0x0001)     |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Status (TLV)                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

Status TLV
  Indicates the event being signalled. The encoding for the Status TLV is specified in Section "Status TLV".

Optional Parameters
This variable length field contains 0 or more parameters, each encoded as a TLV. The following Optional Parameters are generic and may appear in any Notification Message:

Optional Parameter	Type	Length	Value
Extended Status	0x0301	4	See below

Other Optional Parameters, specific to the particular event being signalled by the Notification Messages may appear. These are described elsewhere.

Extended Status
The 4 octet value is an Extended Status Code that encodes additional information that supplements the status information contained in the Notification Status Code.

3.4.1.1. Notification Message Procedures

If an LSR encounters a condition requiring it to notify its peer with advisory or error information it sends the peer a Notification message containing a Status TLV that encodes the information and optionally additional TLVs that provide more information about the event.

If the condition is one that is a fatal error the Status Code carried in the notification will indicate that. In this case, after sending the Notification message the LSR should terminate the LDP session by closing the session TCP connection and discard all state associated with the session, including all label-FEC bindings learned via the session.

When an LSR receives a Notification message that carries a Status Code that indicates a fatal error, it should terminate the LDP session immediately by closing the session TCP connection and discard all state associated with the session, including all label-FEC bindings learned via the session.

3.4.1.2. Events Signalled by Notification Messages

It is useful for descriptive purpose to classify events signalled by Notification Messages into the following categories.

3.4.1.2.1. Malformed PDU or Message

Malformed LDP PDUs or Messages that are part of the LDP Discovery mechanism are handled by silently discarding them.

An LDP PDU received on a TCP connection for an LDP session is malformed if:

- The LDP Identifier in the PDU header is unknown to the receiver, or it is known but is not the LDP Identifier associated by the receiver with the LDP session. This is a fatal error signalled by the Bad LDP Identifier Status Code.

- The LDP protocol version is not supported by the receiver, or it is supported but is not the version negotiated for the session during session establishment. This is a fatal error signalled by the Bad Protocol Version Status Code.

- The PDU Length field is too short (< 20) or too long (> TBD). This is a fatal error signaled by the Bad PDU Length Status Code.

An LDP Message is malformed if:

- The Message Type is unknown. See Section "Unknown Message Types" for more detail.

  If the Message Type is < 0x80000000 (high order bit = 0) it is a fatal error signalled by the Unknown Message Type Status Code.

  If the Message Type is >= 0x8000000 (high order bit = 1) it is silently discarded.

- The Message Length is too large, that is, indicates that the message extends beyond the end of the containing LDP PDU. This is a fatal error signalled by the Bad Message Length Status Code.

3.4.1.2.2. Unknown or Malformed TLV

Malformed TLVs contained in LDP messages that are part of the LDP Discovery mechanism are handled by silently discarding the containing message.

A TLV contained in an LDP message received on a TCP connection of an LDP is malformed if:

- The TLV Length is too large, that is, indicates that the TLV extends beyond the end of the containing message. This is a fatal error signalled by the Bad TLV Length Status Code.

- The TLV type is unknown. See Section "Unknown TLV in Known Message Type" for more detail.

  If the TLV type is < 0x80000000 (high order bit 0) it is a fatal error signalled by the Unknown TLV Status Code.

  If the TLV type is >= 0800000000 (high order bit 1) the TLV is silently dropped. Section "Unknown TLV in Known Message Type" elaborates on this behavior.

- The TLV Value is malformed. This occurs when the receiver handles the TLV but cannot decode the TLV Value. This is intrepreted as indicative of a bug in either the sending or receiving LSR. It is a fatal error signalled by the Malformed TLV Value Status Code.

3.4.1.2.3. Session Hold Timer Expiration

This is a fatal error signalled by the Hold Timer Expired Status Code.

3.4.1.2.4. Unilateral Session Shutdown

This is a non-fatal event signalled by the Shutdown Status Code. The Notification Message may optionally include an Extended Status TLV to provide a reason for the Shutdown. Note that although this is a "non-fatal" event, the sending LSR terminates the session immediately after sending the Notification.

3.4.1.2.5. Initialization Message Events

The session initialization negotiation (see Section "Session Initialization") may fail if the session parameters received in the Initialization Message are unacceptable. This is a fatal error. The specific Status Code depends on the parameter deemed unacceptable, and are defined in Sections "Initialization Message Notification Status Codes".

3.4.1.2.6. Events Resulting From Other Messages

Messages other than the Initialization message may result in events that must be signalled to LDP peers via Notification Messages. These events and the Status Codes used in the Notification Messages to signal them are described in the sections that describe these messages.

3.4.1.2.7. Explicitly Routed LSP Setup Events

Establishment of an Explicitly Routed LSP may fail for a variety of reasons. All such failures are considered non-fatal conditions and they are signalled by the Explicit Response Message.

3.4.1.2.8. Miscellaneous Events

These are events that fall into none of the categories above. There are no miscellaneous events defined in this version of the protocol.

3.4.2. Hello Message

LDP Hello Messages are exchanged as part of the LDP Discovery Mechanism; see Section "LDP Discovery".

The encoding for the Hello Message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Hello (0x0100)            |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

Optional Parameters
  This variable length field contains 0 or more parameters, each encoded as a TLV. The optional parameters defined by this version of the protocol are

Optional Parameter	Type	Length	Value
Targeted Hello	0x0400	0	--
Send Targeted Hello	0x0401	0	--
Transport Address	0x0402	4	See below
Hello Hold Time	0x0403	4	See below

Targeted Hello
This Hello is a Targeted Hello. Without this optional parameter the Hello is a Link Hello.

Send Targeted Hello
Requests the receiver to send periodic Targeted Hellos to the source of this Hello. An LSR initiating Extended Discovery uses this option.

Transport Address
Specifies the IPv4 address to be used for the sending LSR when opening the LDP session TCP connection. If this optional TLV is not present the IPv4 source address for the UDP packet carrying the Hello should be used.

Hello Hold Time
An LSR maintains a record of Hellos received from potential peers (see below) When present, this parameter specifies the time in seconds the sending LSR will maintain its record of Hellos from the receiving LSR without receipt of another Hello. When not present, the sender will use a default hold time. There are interface type specific defaults for Link Hellos as well a default for Targeted Hellos.

3.4.2.1. Hello Message Procedures

An LSR receiving Hellos from another LSR maintains a Hello adjacency for the Hellos. The LSR maintains a hold timer with the Hello adjacency which it restarts whenever it receives a Hello that matches the Hello adjacency. If the hold timer for a Hello adjacency expires the LSR discards the Hello adjacency: see sections "Maintaining Hello Adjacencies" and "Maintaining LDP Sessions".

A LSR processes a received LDP Hello as follows:

1. The LSR checks whether the Hello is acceptable. The criteria for determining whether a Hello is acceptable are implementation dependent (see below for example criteria).

2. If the Hello is not acceptable, the LSR ignores it.

3. If the Hello is acceptable, the LSR checks whether it has a Hello adjacency for the Hello source. If so, it restarts the hold timer for the Hello adjacency. If not it creates a Hello adjacency for the Hello source and starts its hold timer.

4. If the Hello carries any optional TLVs the LSR processes them (see below).

5. Finally, if the LSR has no LDP session for the label space specified by the LDP identifier in the common header for the Hello, it attempts to establish a session for the label space; see section "LDP Session Establishment".

The following are examples of acceptability criteria for Link and Targeted Hellos:

A Link Hello is acceptable if the interface on which it was received has been configured for label switching.

A Targeted Hello from IP source address a.b.c.d is acceptable if either:

- The LSR has been configured to accept Targeted Hellos, or

- The LSR has been configured to send Targeted Hellos to a.b.c.d.

The following describes how an LSR processes Hello optional TLVs:

Targeted Hello
No special processing required.

Send Targeted Hello
If the Send Targeted Hello option is carried by the Hello, the LSR checks whether it has been configured to send Targeted Hellos to the Hello source in response to Hellos with this option. If not, it ignores the option. If so, it initiates periodic transmission of Targeted Hellos to the Hello source.

Transport Address
The LSR associates the specified transport address with the

```
             Hello adjacency.

           Hello Hold Time
             A pair of LSRs negotiate the hold times they use for Hellos
             from each other.  Each LSR proposes a hold time in its Hel-
             los either explicitly by including the Hold Time optional
             TLV or implicitly by omitting it.  The hold time used by
             the LSRs is the minimum of the hold times proposed in their
             Hellos.

       We recommend that the interval between Hello transmissions be at
       most one third of the Hello hold time.
```

3.4.3. Initialization Message

The LDP Initialization Message is exchanged as part of the LDP session establishment procedure; see Section "LDP Session Establishment".

The encoding for the Initialization Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Initialization (0x0200)    |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Common Session Parameters TLV             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Message Id
     Four octet integer used to identify this message.

   Common Session Parameters TLV
     Specifies values proposed by the sending LSR for parameters common
     to all LDP sessions.

     The encoding for the Basic Session Parameters TLV is:
```

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | Common Sess Params (0x0500)  |      Message Length            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | Protocol Version              |      Hold Time                |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                Receiver LDP Identifer                         |
       +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

       Protocol Version
         Two octet unsigned integer containing the version number of the
         protocol. This version of the specification specifies LDP pro-
         tocol version 1.

       Hold Time
         Two octet unsigned non zero integer that indicates the number
         of seconds that the sending LSR proposes for the value of the
         KeepAlive Interval. The receiving LSR MUST calculate the value
         of the KeepAlive Timer by using the smaller of its proposed
         Hold Time and the Hold Time received in the PDU. The value
         chosen for Hold Time indicates the maximum number of seconds
         that may elapse between the receipt of successive PDUs from the
         LSR peer. The Keepalive Timer is reset each time a PDU
         arrives.

       Receiver LDP Identifer
         Identifies the receiver's label space. This LDP Identifier,
         together with the sender's LDP Identifier in the common header
         enables the receiver to match the Initialization message with
         one of its Hello adjacencies; see Section "Hello Message Pro-
         cedures".

    Optional Parameters
      This variable length field contains 0 or more parameters, each
      encoded as a TLV. The optional parameters are:

Optional Parameter	Type	Length	Value
Label Allocation Discipline	0x0501	1	See below
Loop Detection	0x0502	0	--
Merge	0x0503	1	See below
ATM Null Encapsulation	0x0504	0	--
ATM Label Range	0x0600	8	See below
Frame Relay Label Range	0x0601	8	See below

Label Allocation Discipline

Indicates the type of Label allocation.   A value of 0 is Downstream allocation, A value of 1 is Downstream On Demand. If this optional parameter is not specfied, Downstream allocation is used.

Loop Detection

If present, indicates that Loop Detection is enabled. If absent, Loop Detection is disabled.

Merge

Specifies the merge capabilities of an ATM or Frame Relay switch. The following values are supported in this version of the specification:

Value	Meaning
0	Merge not supported
For ATM Merge:	
1	VP Merge supported
2	VC Merge supported
3	VP & VC Merge supported
For Frame Relay Merge:	
Non-zero	Merge supported

ATM Null Encapsulation

If present, specifies that the LSR supports the null encapsulation of [rfc1483] for its data VCs on the ATM link managed by the LDP session. In this case IP packets are carried directly inside AAL5 frames. If absent, the null encapsulation is not supported.

ATM Label Range

Used when an LDP session manages label exchange for an ATM link. The ATM Label Range TLV contains the label range supported by the transmitting LSR. A receiving LSR MUST calculate the intersection between the received range and its own supported label range. The intersection is the range in which the LSR may allocate and accept labels. LSRs may NOT establish an adjacency with neighbors whose intersection range is NULL.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Res  |    Minimum VPI        |        Minimum VCI            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Res  |    Maximum VPI        |        Maximum VCI            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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 should be right justified in this field and preceding bits should be set to 0.

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 should be right justified in this field and preceding bits should be set to 0.

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 should be right justified in this field and preceding bits should be set to 0.

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 should be right justified in this field and preceding bits should be set to 0.

```
       Frame Relay Label Range
         Used when an LDP session manages label exchange for a Frame
         Relay link.  The Frame Relay Label Range TLV contains the label
         range supported by the transmitting LSR.  A receiving LSR MUST
         calculate the intersection between the received range and its
         own supported label range.  The intersection is the range in
         which the LSR may allocate and accept labels.  LSRs may NOT
         establish an adjacency with neighbors whose intersection range
         is NULL.

          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
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | Reserved        |Len|                     Minimum DLCI        |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | Reserved        |                         Maximum DLCI        |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


         Len
           This field specifies the number of bits of the DLCI.  The
           following values are supported:

                Len    DLCI bits

                0       10
                1       17
                2       23
```

3.4.3.1. Initialization Message Procedures

See Section "LDP Session Establishment" and particularly Section "Session Initialization" for general procedures for handling the Initialization Message.

3.4.4. KeepAlive Message

An LSR sends KeepAlive Messages as part of a mechanism that monitors the integrity of the LDP session transport connection.

The encoding for the KeepAlive Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     KeepAlive (0x0201)        |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

Optional Parameters
  No optional parameters are defined for the KeepAlive message.

3.4.4.1. KeepAlive Message Procedures

The Hold Timer mechanism described in Section "Maintaining LDP Ses-
sions" resets a seesion hold timer every time an LDP PDU is received.
The KeepAlive Message is provided to allow reset of the Hold Timer in
circumstances where an LSR has no other information to communicate to
an LDP peer.

An LSR must arrange that its peer sees an LDP Message from it at
least every Hold Time period. That message may be any other from the
protocol or, in circumstances where there is no need to send one of
them, it must be KeepAlive Message.

3.4.5. Address Message

An LSR sends the Address Message to an LDP peer to advertise its
interface addresses.

The encoding for the Address Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address (0x0300)          |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                     Address List TLV                          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

Address List TLV
  The list of interface addresses being advertised by the sending LSR. The encoding for the Address List TLV is specified in Section "Address List TLV".

Optional Parameters
  No optional parameters are defined for the Address message.

3.4.5.1. Address Message Procedures

An LSR that receives an Address Message message uses the addresses it learns to maintain a database for mapping between peer LDP Identifiers and next hop addresses; see section "LDP Identifiers and Next Hop Addresses".

When a new LDP session is initialized and before sending Label Mapping or Label Request messages and LSR should advertise its interface addresses with one or more Address messages.

Whenever an LSR "activates" a new interface address, it should advertise the new address with an Address message.

Whenever an LSR "de-activates" a previously advertised address, it should withdraw the address with an Address Withdraw message; see Section "Address Withdraw Message".

3.4.6. Address Withdraw Message

An LSR sends the Address Message to an LDP peer to withdraw previously advertised interface addresses.

The encoding for the Address Withdraw Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Address Withdraw (0x0301) |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                     Address List TLV                          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

Address list TLV
  The list of interface addresses being withdrawn by the sending LSR. The encoding for the Address list TLV is specified in Section "Address List TLV".

Optional Parameters
  No optional parameters are defined for the Address Withdraw message.

3.4.6.1. Address Withdraw Message Procedures

See Section "Address Message Procedures"

3.4.7. Label Mapping Message

An LSR sends a Label Mapping message to an LDP peer to advertise FEC-label bindings to the peer.

The encoding for the Label Mapping Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Label Mapping (0x0400)    |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC-Label Mapping TLV 1                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC-Label Mapping TLV n                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

FEC-Label Mapping TLV
  Each specifies a binding between an FEC and a label. A FEC-Label
  Mapping TLV is a nested TLV that contains a FEC TLV, a Label TLV,
  an optional COS TLF, an optional Hop Count TLV, and an optional
  Path Vector TLV:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   FEC-label Mapping (0x0700)  |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC TLV                                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Label TLV                                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     COS TLV (optional)                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Hop Count TLV (optional)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Path Vector TLV (optional)                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The encodings for the FEC, Label, COS, Hop Count, and Path Vector
TLVs can be found in Section "Commonly Used TLVs".

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

Need to add multipath possibility to above by allowing multiple label TLVs to the FEC-label Mapping TLV. This will be done with the addition:

```
         Label TLV2 (optional)
         ...
         Label TLVn (optional)
```

with discussion.

END NOTE * END NOTE * END NOTE:

Optional Parameters
  No optional parameters are defined for the Label Mapping message.

3.4.7.1. Label Mapping Message Procedures

The Mapping message is used by an LSR to distribute a label mapping for a FEC to its LDP peers. If an LSR distributes a mapping for a FEC to multiple LDP peers, it is a local matter whether it maps a single label to the FEC, and distributes that mapping to all its peers, or whether it uses a different mapping for each of its peers.

An LSR is always responsible for the consistency of the label mappings it has distributed, and that its peers have these mappings.

3.4.7.1.1. Independent Control Mapping

If an LSR is configured for independent control, a mapping message is transmitted by an LSR to peers upon any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table, and the label advertisement mode is Downstream allocation.

2. The LSR receives a Request message from an upstream peer for an FEC present in the LSR's forwarding table.

3. The next hop for an FEC changes to another LDP peer, and loop detection is configured.

4. The attributes of a mapping change.

5. The receipt of a mapping from the downstream next hop AND
    a) no upstream mapping has been created OR
    b) loop detection is configured OR
    c) the attributes of the mapping have changed.

3.4.7.1.2. Ordered Control Mapping

If an LSR is doing ordered control, a Mapping message is transmitted by downstream LSRs upon any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table, and is the egress for that FEC.

2. The LSR receives a Request message from an upstream peer for an FEC present in the LSR's forwarding table, and the LSR is the egress for that FEC OR has a downstream mapping for that FEC.

3. The next hop for an FEC changes to another LDP peer, and loop detection is configured.

4. The attributes of a mapping change.

5. The receipt of a mapping from the downstream next hop AND
    a) no upstream mapping has been created OR
    b) loop detection is configured OR
    c) the attributes of the mapping have changed.

3.4.7.1.3. Downstream-on-Demand Label Advertisement

In general, the upstream LSR is responsible for requesting label mappings when operating in Downstream-on-Demand mode. However, unless some rules are followed, it is possible for neighboring LSRs with different advertisement modes to get into a livelock situation where everything is functioning properly, but no labels are distributed. For example, consider two LSRs Ru and Rd where Ru is the upstream LSR and Rd is the downstream LSR for a particular FEC. In this example, Ru is using Downstream allocation mode and Rd is using Downstream-on-Demand mode. In this case, Rd may assume that Ru will request a label mapping when it wants one and Ru may assume that Rd will advertise a label if it wants Ru to use one. If Rd and Ru operate as suggested, no labels will be distributed and packets must be routed at layer-3.

This livelock situation can be avoided if the following rule is observed: an LSR operating in Downstream-on-Demand mode should not be expected to send unsolicited mapping advertisements. Therefore, if the downstream LSR is operating in Downstream-on-Demand mode, the upstream LSR is responsible for requesting label mappings as needed. However, if all interfaces on an LSR are configured to operate in Downstream- on-Demand mode the LSR can wait to issue a request until a corresponding request has been sent from an upstream LSR.

3.4.7.1.4. Downstream Allocation Label Advertisement

In general, the downstream LSR is responsible for advertising a label mapping when it wants an upstream LSR to use the label. An upstream LSR may issue a mapping request if it so desires.

3.4.8. Label Request Message

An LSR sends the Label Request Message to an LDP peer to request a binding (mapping) for one or more specific FECs.

The encoding for the Label Request Message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Label Request (0x0401)    |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC-Request TLV 1                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC-Request TLV n                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

FEC-Request TLV

Each specifies an FEC for which a label mapping is requested. A FEC-Request TLV is a nested TLV that contains a FEC TLV, an optional COS TLV, and an optional Hop Count TLV.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   FEC-Request (0x0701)        |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC TLV                                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     COS TLV (optional)                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Hop Count TLV (optional)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The encodings for the FEC, COS, and Hop Count TLVs are specified in Section "Commonly Used TLVs".

Optional Parameters
No optional parameters are defined for the Label Request message.

3.4.8.1. Label Request Message Procedures

The Request message is used by an upstream LSR to explicitly request that the downstream LSR assign and advertise a label for an FEC.

An LSR transmits a Request message under any of the following condi- tions:

1. The LSR recognizes a new FEC via the forwarding table, and the next hop is an Operational LDP peer, and the LSR doesn't already have a mapping from the next hop for the given FEC.

2. The next hop to the FEC changes, and the LSR doesn't already have a mapping from that next hop for the given FEC.

If a request cannot be satisfied by the downstream LSR, the request- ing LSR may optionally choose to request again at a later time, or, if the downstream LSR is configured for Downstream Allo- cation, the requesting LSR may wait for the mapping, assuming that the downstream LSR will provide the mapping automatically when it is available.

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

In the case where the downstream LSR is doing DoD, how does the

requesting LSR decide when to make its request?

TDP addresses this issue by having a "now I have label resources" message which it sends to downwstream peers whose requests it has denied. This serves as a signal to them to re-issue their requests. LDP should probably have this. Without such a signal, the denied requester has no recourse but to periodically retry.

END NOTE * END NOTE * END NOTE:

3.4.9. Label Withdraw Message

An LSR sends a Label Withdraw Message to an LDP peer to signal the peer that the peer may not continue to use specific FEC-label mappings the LSR had previously advertised. This breaks the mapping between the FECs and the labels.

The encoding for the Label Withdraw Message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Label Withdraw (0x0402)   |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message ID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      FEC-Withdraw-Release TLV 1               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      FEC-Withdraw-Release TLV n               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Optional Parameters                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

FEC-Withdraw-Release TLV
  Each TLV specifies a FEC-label mapping being withdrawn. A FEC-Withdraw-Release TLV is a nested TLV that contains a FEC TLV and an optional label TLV.

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | FEC-Withdraw-Release (0x0702) |      Length                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     FEC TLV                                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     Label TLV (optional)                      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The encodings for the FEC and Label TLVs are specified in Section "Commonly Used TLVs".

NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

Need to add multipath possibility to above by allowing multiple label TLVs to the FEC-label Mapping TLV. This will be done with the addition:

```
         Label TLV2 (optional)
         ...
         Label TLVn (optional)
```

with discussion.

END NOTE * END NOTE * END NOTE:

Optional Parameters
No optional parameters are defined for the Label Withdraw message.

3.4.9.1. Label Withdraw Message Procedures

An LSR transmits a Withdraw message under the following condition:

1. The LSR no longer recognizes a previously known FEC.

2. Optionally, the LSR has unspliced an upstream label from the downstream label.

The FEC in the FEC-Withdraw-Release TLV is a FEC for which labels are to be withdrawn. If no label TLV follows the FEC, all labels associ-ated with the FEC are to be withdrawn, else only the labels specified in the following Label TLV are to be withdrawn.

3.4.10. Label Release Message

An LSR sends a Label Release message to an LDP peer to signal the peer that the LSR no longer needs specific FEC-label mappings previously requested of and/or advertised by the peer.

The encoding for the Label Release Message is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Label Release (0x0403)    |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC-Withdraw-Release TLV 1                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC-Withdraw-Release TLV n                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

FEC-Withdraw-Release TLVs
  Each TLV specifies a FEC-label mapping being released. The encoding for the FEC-Withdraw-Release TLV is specified in Section "Withdraw Message".

  NOTE*NOTE*NOTE*NOTE*NOTE*NOTE:

    Need to add multipath possibility to above by allowing multiple label TLVs to the FEC-label Mapping TLV. This will be done with the addition:

```
             Label TLV2 (optional)
             ...
             Label TLVn (optional)
```

    with discussion.

  END NOTE * END NOTE * END NOTE:

Optional Parameters
No optional parameters are defined for the Label Release message.

3.4.10.1. Label Release Message Procedures

An LSR transmits a Release message to a peer when it is no longer needs a label previously received from or requested of that peer.

An LSR transmits a Release message under any of the following conditions:

1. The LSR which sent the label mapping is no longer the next hop for the mapped FEC, and the LSR is configured for conservative operation.

2. The LSR determines that a previously received label is no longer valid, as the downstream LSR from which it was received is no longer the next hop for the FEC, and the LSR is configured for conservative operation.

3. The LSR has received a Withdraw message for a previously received label.

Note that if an LSR is configured for "liberal mode", a release message will never be transmitted in the case of conditions (1) and (2) as specified above. In this case, the upstream LSR keeps each unused label, so that it can immediately be used later if the downstream peer becomes the next hop for the FEC.

The FEC in the FEC-Withdraw-Release TLV is a FEC for which labels are to be released. If no label TLV follows the FEC TLV, all labels associated with the FEC are to be released, else only the labels specified in the following Label TLV are to be released.

3.4.11. Label Query Message

An LSR sends a Label Query message to an LDP peer when performing the loop prevention diffusion algorithm on an FEC.

The encoding for the Label Query Message is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Label Query (0x0405)      |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC TLV                                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Path Vector TLV                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Id
  Four octet integer used to identify this message.

The encodings for the FEC and Path Vector TLVs can be found in Section "Commonly Used TLVs".

Optional Parameters
  No optional parameters are defined for the Label Query message.

3.4.11.1. Label Query Message Procecures

See Section "Loop Prevention via Diffusion" for general procedures for handling the Query Message.

3.4.12. Explicit Route Request Message

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       ER Request (0x0500)     |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Message ID                                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC-ER TLV 1                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC-ER TLV n                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Message Id
     Four octet integer used to identify this message.

   FEC-ER TLV
     Each specifies a binding between an FEC and a label.  A FEC-ER TLV
     is a nested TLV that contains a FEC TLV, a Label TLV, an explicit-
     route identifier (ERLSPID) TLV, the explict-route TLV, an optional
     COS TLF, and an optional Bandwith Reservation TLV:

      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |         FEC-ER TLV  (0x0703)  |      Length                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    FEC TLV                                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    ERLSPID TLV                                |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    Explicit Route TLV                         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    COS TLV (optional)                         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    Bandwidth Reservation TLV (optional)       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


     The encodings for the FEC and COS TLVs can be found in Section
     "Commonly Used TLVs".

     ERLSPID TLV
       The globally unique value that identifies the explicit route.
       The encoding for the ERLSPID is:

        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |         ERLSPID (0x0801)      |      Length                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       Explicit Identifier                     |
       +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                               |                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
       |                     Peg Explicit Identifier                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


       Explicit Identifier
         A 6-octet globally unique value that identifies the explicit
```

route LSP. It is generated by the LSR that creates the Explicit Request message. The first four octets is the LSR IP Address. The last two octets contain a `Local identifier' value. It is incumbent on an LSR that originates an Explicit Request message to choose an unused value for the Local Identifier.

Peg Explicit Identifier

A 6-octet globally unique value that identifies a loose segment of an explicit route LSP. It is generated by the upstream peg LSR that creates the loose segment. The first four octets is the LSR IP Address. The last two octets contain a 'Local identifier' value. It is incumbent on a peg LSR that creates a loose segment to choose an unused value for the Local Identifier every time the segment is reestablished. When a segment is strictly routed this field is set to zero by the sender and ignored by the receiver.

Explicit Route TLV

The sequence of ER Next Hop (ERNH) TLVs and a pointer to the one that should be processed by the LSR that receives this ER TLV. The encoding for the Explicit Route is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Explicit Route TLV  (0x0800) |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Next ERNH TLV Pointer     |     Reserved        |P|Preempt|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  ERNH TLV  (Variable length)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next ERNH TLV Pointer

This 16 bit unsigned integer points to the offset in octets of the next ERNH TLV to be processed. The first octet after the two reserved octets that follow this pointer is defined to have an offset value of zero. For example an ERNH TLV Pointer value of zero would point to the first ERNH TLV in the sequence of ERNH Objects.

P bit

when set indicates that the loosely routed segments must remain pinned-down. ERLSP must be rerouted only when adjacency is lost along the segment. When not set indicates loose segment is not pinned down and must be changed to match the underlying hop-by-hop path.

```
       Preempt
         A 16 level preemption is provided to facilitate placement of
         ERLSP when resources aren't available.  Each LSR maintains this
         value in the ERLSP control block.  A higher preemption value
         can preempt LSPs with lower value.

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

       ERNH TLV
         This TLV contains the four octet IP address of an LSR through
         which the Explicit Route LSP is to pass and an (optional)
         reservation (RES) TLV to be processed by that LSR.

         The strict TLV indicates that the ER LSP setup must be routed
         directly via the LSR indicated in the ERNH object; i.e. that
         that LSR must be the next hop in the Explicit Route LSP's path.
         The loose TLV indicates that the LSP may be routed in any way;
         i.e. via other unspecified LSRs, so long as it (eventually)
         reaches the LSR specified in the ERNH object.  This TLV may be
         followed by the optional Reservation TLV.

         The ERNH encodings are:
          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
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |      ER Strict TLV  (0x0802) |      Length                   |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                             IPv4 Address                     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


          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
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |      ER Loose TLV  (0x0803)   |      Length                  |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                             IPv4 Address                     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


       Ipv4 Address
         The IP address of the next LSR in the Explicit Route LSP.

     Bandwidth Reservation TLV
       Specifies the bandwidth reservation required at each LSR hop.
       The encoding for the Bandwidth Reservation is:
```

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |      Bandwidth TLV  (0x0804)  |      Length                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                      BW requirement                           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

     BW Requirement
       Unsigned 32 bit integer representing the bandwidth, in units of
       kilo bps, that must be reserved for the LSP at every LSR identi-
       fied in the ERNH Object. The bandwidth is guaranteed within a
       coarser time period allowing for simpler implementations. The
       specified bandwidth is guaranteed within several milliseconds or
       a few seconds time period. Nodes may also use this as a minimal
       bandwidth guarantee within the same time period.

3.4.12.1. Explicit Route Request Procedures

See Sections "Explicitly Routing LSPs" and "ERLSP State Machine" for general procedures for handling the Explicit Route Request Message.

3.4.13. Explicit Route Response Message

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      ER Response (0x0501)     |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Message ID                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      ERLSPID TLV                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Label TLV                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Status TLV                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The encodings for the Label, and Status TLVs can be found in Section 3.3.3 ("Commonly Used TLVs").

Message Id
  Four octet integer used to identify this message.

   ERLSPID TLV
     The globally unique value used for ERLSPID in the Explicit Request
     message that elicited this Response message. The encoding for the
     ERLSPID (shown above and repeated here for convenience) is:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         ERLSPID (0x0801)      |      Length                   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Explicit Identifier                     |
      +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                               |                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
      |                     Peg Explicit Identifier                   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

     Explicit Identifier
       A 6-octet globally unique value that identifies the explicit
       route LSP. It is generated by the LSR that creates the Explicit
       Request message. The first four octets is the LSR IP Address.
       The last two octets contain a `Local identifier' value. It is
       incumbent on an LSR that originates an Explicit Request message
       to choose an unused value for the Local Identifier.

     Peg Explicit Identifier
       A 6-octet globally unique value that identifies a loose segment
       of an explicit route LSP. It is generated by the upstream peg
       LSR that creates the loose segment. The first four octets is the
       LSR IP Address. The last two octets contain a 'Local identifier'
       value. It is incumbent on a peg LSR that creates a loose segment
       to choose an unused value for the Local Identifier every time the
       segment is reestablished. When a segment is strictly routed this
       field is set to zero by the sender and ignored by the receiver.

3.4.13.1. Explicit Route Response Procedures

   See Sections "Explicitly Routing LSPs" and "ERLSP State Machine" for
   general procedures for handling the Explicit Response Request Mes-
   sage.

3.5. Messages and TLVs for Extensibility

The procedures to provide for LDP extensiblity include rules for handling unknown messages and TLVs. The rules described in the sections that follow make use of the high order bits in the message or TLV type field. In these rules, "b" represents an arbitray bit value in a message or TLV type.

3.5.1. Procedures for Unknown Messages and TLVs

3.5.1.1. Unknown Message Types

When a message with an unknown Message Type is received, there are two possibilities as described below. The choice for how to handle an unknown Message Type is determined by the high-order bit of the Message Type field.

- Message Type = 0bbbbbbbbbbbbbbb

  The entire message must be rejected and the event signalled by a Notification Message with the Unknown Message Type Status Code.

- Message Type = 1bbbbbbbbbbbbbbb

  The entire message must be dropped silently (i.e., it should be ignored and no error should be returned).

  In either case described above, an LSR that does not understand the message type must not attempt to process the message.

3.5.1.2. Unknown TLV in Known Message Type

When an unknown TLV is found in a known Message Type, there are three possibilities as described below. The choice for how to handle an unknown TLV is determined by the high-order two bits of the TLV Type field.

- TLV Type = 0bbbbbbbbbbbbbbb

  The entire message must be rejected and the event signalled by a Notification Message with the Unknown TLV Status Code.

- TLV Type = 10bbbbbbbbbbbbbb

  The TLV must be dropped silently (i.e., it should be ignored and no error should be returned). If the semantics of the including

Message Type dictate that message be forwarded to other nodes, the TLV must not be forwarded with the message.

- TLV Type = 11bbbbbbbbbbbbbb

  The TLV must be silently ignored (i.e., no error should be returned). If the semantics of the including Message Type dictate that message be forwarded to other nodes, the TLV must be forwarded unmodified with the message.

3.5.2. LDP Vendor-Private Extensions

Both Vendor-Private Messages and Vendor-Private Objects are defined to convey vendor-private information or LDP extensions between LDP nodes. These extensions may also be useful for experimentation in existing networks.

3.5.2.1. LDP Vendor-Private TLV

The following three Vendor-Private TLV classes are defined to be used in any message:

- Vendor Private TLV Class 1. TLV type values:

  0x3FXX (boolean 00111111bbbbbbbb)

- Vendor Private TLV Class 2. TLV type values:

  0xBFXX (boolean 10111111bbbbbbbb)

- Vendor Private TLV Class 3, TLV type values:

  0xFFXX (boolean 11111111bbbbbbbb)

These TLVs are to be handled according to the high order bit(s) of the TLV type. The unspecified part of the TLV type is assigned by the vendor and should be interpreted by a receiving LSR only if it understands the Vendor ID encoded in the TLV Value field.

The Value field of a Vendor Private TLV is defined as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Vendor ID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Data....                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Vendor Id
802 Vendor ID as assigned by the IEEE.

Data
The remaining octets after the Vendor ID in the Value field are optional vendor-dependent data.

3.5.2.2. LDP Vendor-Private Messages

The LDP Vendor-Private Message is carried in LDP PDUs to convey vendor-private information or LDP extensions between LSRs.

The following two Vendor-Private Message classes are defined:

- Vendor Private Message Class 1. Message type values:

  0x7FXX (boolean 01111111bbbbbbbb)

- Vendor Private Message Class 2. Message type values:

  0xFFXX (boolean 11111111bbbbbbbb)

The first TLV in a vendor private message must be the Vendor Private ID TLV, a Vendor Private Class 3 TLV, encoded as shown below:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0xFF     |      0x00     |             0x04              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Vendor ID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Vendor-Private messages are to be handled according to the high order bit of the message type number. The determination as to

whether the Vendor-Private message is understood is based on the Vendor ID in first TLV in the message body.

3.6. TLV Summary

The following are the TLVs defined in this version of the protocol.

TLV	Type	Section Title
FEC	0x0100	"FEC TLV"
Address List	0x0101	"Address List TLV"
COS	0x0102	"COS TLV"
Hop Count	0x0103	"Hop Count TLV"
Path Vector	0x0104	"Path Vector TLV"
Generic Label	0x0200	"Generic Label TLV"
ATM Label	0x0201	"ATM Label TLV"
Frame Relay Label	0x0202	"Frame Relay Label TLV"
Status	0x0300	"Status TLV"
Extended Status	0x0301	"Notification Message"
Targeted Hello	0x0400	"Hello Message"
Send Targeted Hello	0x0401	"Hello Message"
Transport Address	0x0402	"Hello Message"
Hello Hold Time	0x0403	"Hello Message"
Common Session Parameters	0x0500	"Initialization Message"
Label Allocation Discipline	0x0501	"Initialization Message"
Loop Detection	0x0502	"Initialization Message"
Merge	0x0503	"Initialization Message"
ATM Null Encapsulation	0x0504	"Initialization Message"
ATM Label Range	0x0600	"Initialization Message"
Frame Relay Label Range	0x0601	"Initialization Message"
FEC-Label Mapping	0x0700	"Label Mapping Message"
FEC-Request	0x0701	"Label Request Message"
FEC-Withdraw-Release	0x0702	"Label Withdraw Message"
FEC-ER TLV	0x0703	"Explicit Request Message"
Explicit Route	0x0800	"Explicit Request Message"
ERLSPID	0x0801	"Explicit Request Message"
ER Strict	0x0802	"Explicit Request Message"
ER Loose	0x0803	"Explicit Request Message"
Bandwidth	0x0804	"Explicit Request Message"

3.7. Status Code Summary

The following are the Status Codes defined in this version of the protocol.

Status Code	Type	Section Title
Success	0x0000	"Status TLV"
Bad LDP Identifer	0x8001	"Events Signalled by ..."
Bad Protocol Version	0x8002	"Events Signalled by ..."
Bad PDU Length	0x8003	"Events Signalled by ..."
Unknown Message Type	0x8004	"Events Signalled by ..."
Bad Message Length	0x8005	"Events Signalled by ..."
Unknown TLV	0x8006	"Events Signalled by ..."
Bad TLV length	0x8007	"Events Signalled by ..."
Malformed TLV Value	0x8008	"Events Signalled by ..."
Hold Timer Expired	0x8009	"Events Signalled by ..."
Shutdown	0x000A	"Events Signalled by ..."
Loop Detected	0x000B	"Loop Detection Via Diffusion"

4. Security

Security considerations will be addressed in a future revision of this document.

5. Acknowledgments

The ideas and text in this document have been collected from a number of sources. We would like to thank Rick Boivie, Ross Callon, Alex Conta, Eric Rosen, Bernard Suter, Yakov Rekhter, and Arun Viswanathan.

6. References

[FRAMEWORK] Callon et al, "A Framework for Multiprotocol Label Switching" draft-ietf-mpls-framework-01.txt, July 1997

[ARCH] Rosen et al, "A Proposed Architecture for MPLS" draft-ietf-mpls-arch-02.txt, July 1998

[ENCAP] Farinacci et al, "MPLS Label Stack Encoding" draft-ietf-mpls-label-encaps-02.txt, July, 1998

[FR] Conta et al, "Use of Label Switching on Frame Relay Networks"

draft-ietf-mpls-fr-01.txt, August, 1998

[rfc1583] J. Moy, "OSPF Version 2", RFC 1583, Proteon Inc, March 1994

[rfc1771] Y. Rekhter, T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, IBM Corp, Cisco Systems, March 1995

[rfc1483] J. Heinanen, "Multiprotocol Encapsulation over ATM Adaptation Layer 5", RFC 1483, Telecom Finland, July 1993

7. Author Information

Loa Andersson
Bay Networks Inc
3 Federal Street
Billerica, MA 01821
email: Loa_Andersson@baynetworks.com

Paul Doolan
Ennovate Networks
330 Codman Hill Rd
Marlborough MA 01719
Phone: 978-263-2002
email: pdoolan@ennovatenetworks.com

Nancy Feldman
IBM Corp.
17 Skyline Drive
Hawthorne NY 10532
Phone: 914-784-3254
email: nkf@us.ibm.com

Andre Fredette
Bay Networks Inc
3 Federal Street
Billerica, MA 01821
Phone: 978-916-8524
email: fredette@baynetworks.com

Bob Thomas
Cisco Systems, Inc.
250 Apollo Dr.
Chelmsford, MA 01824
Phone: 978-244-8078
email: rhthomas@cisco.com