Network Working Group
Internet Draft
Expiration Date: December 2000

Loa Andersson
Nortel Networks Inc.

Paul Doolan
Ennovate Networks

Nancy Feldman
IBM Corp

Andre Fredette
Nortel Networks Inc.

Bob Thomas
Cisco Systems, Inc.

June 2000

LDP Specification

draft-ietf-mpls-ldp-08.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 a set of procedures, called a label distribution protocol, by which one LSR informs another of label bindings it has made. This document defines a set of such procedures called LDP (for Label Distribution Protocol) by which LSRs distribute labels to support MPLS forwarding along normally routed paths [LDPAPPLIC].

Table of Contents

1. LDP Overview

The MPLS architecture [ARCH] defines a label distribution protocol as a set of procedures by which one Label Switched Router (LSR) informs another of the meaning of labels used to forward traffic between and through them.

The MPLS architecture does not assume a single label distribution protocol. In fact, a number of different label distribution protocols are being standardized. Existing protocols have been extended so that label distribution can be piggybacked on them. New protocols have also been defined for the explicit purpose of distributing labels. The MPLS architecture discusses some of the considerations when choosing a label distribution protocol for use in particular MPLS applications such as Traffic Engineering [TE].

The Label Distribution Protocol (LDP) defined in this document is a new protocol defined for distributing labels. It 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.

More information about the applicability of LDP can be found in [LDPAPPLIC].

This document assumes familiarity with the MPLS architecture [ARCH]. Note that [ARCH] includes a glossary of MPLS terminology, such as ingress, label switched path, etc.

1.1. LDP Peers

Two LSRs which use LDP to exchange label/FEC 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 session 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, maintain, and terminate sessions between LDP 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 error information.

Discovery messages provide a mechanism whereby LSRs indicate their presence in a network by sending a Hello message periodically. This is transmitted as a UDP packet to the LDP port at the `all routers on this subnet' group multicast address. When an LSR chooses to establish a session with another 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 messages. To satisfy these requirements LDP uses the TCP transport for session, advertisement and notification messages; i.e., for everything but the UDP-based discovery mechanism.

1.3. LDP Message Structure

All LDP messages have a common structure that uses a Type-Length-Value (TLV) encoding scheme; see Section "Type-Length-Value" encoding. The Value part of a TLV-encoded object, or TLV for short, may itself contain one or more TLVs.

1.4. LDP Error Handling

LDP errors and other events of interest are signaled to an LDP 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 from a peer for an LDP session, it terminates the LDP 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.

1.5. LDP Extensibility and Future Compatibility

Functionality may be added to LDP in the future. It is likely that future 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.

1.6. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [rfc2119].

2. LDP Operation

2.1. FECs

It is necessary to precisely specify which packets may be mapped to each LSP. This is done by providing a FEC specification for each LSP. The FEC identifies the set of IP packets which may be mapped to that LSP.

Each FEC is specified as a set of one or more FEC elements. Each FEC element identifies a set of packets which may be mapped to the

corresponding LSP. When an LSP is shared by multiple FEC elements, that LSP is terminated at (or before) the node where the FEC elements can no longer share the same path.

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

1. Address Prefix. This element is an address prefix of any length from 0 to a full address, inclusive.

2. Host Address. This element is a full host address.

(We will see below that an Address Prefix FEC element which is a full address has a different effect than a Host Address FEC element which has the same address.)

We say that a particular address "matches" a particular address prefix if and only if that address begins with that prefix. We also say that a particular packet matches a particular LSP if and only if that LSP has an Address Prefix FEC element which matches the packet's destination address. With respect to a particular packet and a particular LSP, we refer to any Address Prefix FEC element which matches the packet as the "matching prefix".

The procedure for mapping a particular packet to a particular LSP uses the following rules. Each rule is applied in turn until the packet can be mapped to an LSP.

- If there is exactly one LSP which has a Host Address FEC element that is identical to the packet's destination address, then the packet is mapped to that LSP.

- If there are multiple LSPs, each containing a Host Address FEC element that is identical to the packet's destination address, then the packet is mapped to one of those LSPs. The procedure for selecting one of those LSPs is beyond the scope of this document.

- If a packet matches exactly one LSP, the packet is mapped to that LSP.

- If a packet matches multiple LSPs, it is mapped to the LSP whose matching prefix is the longest. If there is no one LSP whose matching prefix is longest, the packet is mapped to one from the set of LSPs whose matching prefix is longer than the others. The procedure for selecting one of those LSPs is beyond the scope of this document.

- If it is known that a packet must traverse a particular egress router, and there is an LSP which has an Address Prefix FEC element which is an address of that router, then the packet is mapped to that LSP. The procedure for obtaining this knowledge is beyond the scope of this document.

The procedure for determining that a packet must traverse a particular egress router is beyond the scope of this document. (As an example, if one is running a link state routing algorithm, it may be possible to obtain this information from the link state data base. As another example, if one is running BGP, it may be possible to obtain this information from the BGP next hop attribute of the packet's route.)

It is worth pointing out a few consequences of these rules:

- A packet may be sent on the LSP whose Address Prefix FEC element is the address of the packet's egress router ONLY if there is no LSP matching the packet's destination address.

- A packet may match two LSPs, one with a Host Address FEC element and one with an Address Prefix FEC element. In this case, the packet is always assigned to the former.

- A packet which does not match a particular Host Address FEC element may not be sent on the corresponding LSP, even if the Host Address FEC element identifies the packet's egress router.

2.2. Label Spaces, Identifiers, Sessions and Transport

2.2.1. Label Spaces

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 that uses VCIs as labels, or a Frame Relay interface that 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.

2.2.2. LDP Identifiers

An LDP identifier is a six octet quantity used to identify an LSR label space. The first four octets identify the LSR and must be a globally unique value, such as a 32-bit router Id assigned to the LSR. 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:

```
          <LSR Id> : <label space id>
```

e.g., lsr171:0, lsr19:2.

Note that an LSR that manages and advertises multiple label spaces 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 labels) and the other of which is ATM.

2.2.3. LDP Sessions

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

When an LSR uses LDP to advertise more than one label space to another LSR it uses a separate LDP session for each label space.

2.2.4. LDP Transport

LDP uses TCP as a reliable transport for sessions.

When multiple LDP sessions are required between two LSRs there is one TCP session for each LDP session.

2.3. 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 LSRa sends traffic matching some criteria via an LSP to non-directly connected LSRb rather than forwarding the traffic along its normally routed path.

The path between LSRa and LSRb would include one or more intermediate LSRs (LSR1,...LSRn). An LDP session between LSRa and LSRb would enable LSRb to label switch traffic arriving on the LSP from LSRa by providing LSRb means to advertise labels for this purpose to LSRa.

In this situation LSRa would apply two labels to traffic it forwards on the LSP to LSRb: a label learned from LSR1 to forward traffic along the LSP path from LSRa to LSRb; and a label learned from LSRb to enable LSRb to label switch traffic arriving on the LSP.

LSRa first adds the label learned via its LDP session with LSRb 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 learned from LSR1 onto the label stack.

2.4. 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.

2.4.1. 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 on this subnet" 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.

2.4.2. 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 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 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.

2.5. Establishing and Maintaining LDP Sessions

2.5.1. 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.

2.5.2. Transport Connection Establishment

The exchange of Hellos results in the creation of a Hello adjacency at LSR1 that serves to bind 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 a TCP connection for a new LDP 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 (TLV) in Hello's it sends to LSR2 to advertise 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 address used in Hellos it sends to LSR2.

   Similarly, address A2 is determined as follows:

   a. If LSR2 uses the Transport Address optional object, 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 address in Hellos received 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.

The procedure for comparing A1 and A2 as unsigned integers is:

- If A1 and A2 are not in the same address family, they are incomparable, and no session can be established.

- Let U1 be the abstract unsigned integer obtained by treating A1 as a sequence of bytes, where the byte which appears earliest in the message is the most significant byte of the integer and the byte which appears latest in the message is the least significant byte of the integer.

  Let U2 be the abstract unsigned integer obtained from A2 in a similar manner.

- Compare U1 with U2. If U1 > U2, then A1 > A2; if U1 < U2, then A1 < A2.

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.

2.5.3. 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.

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 LDP Initialization message on the connection. The Initialization message carries both the LDP Identifier for the sender's (active LSR's) label space and the LDP Identifier for the receiver's (passive LSR's) label space.

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 PDU header for the Initialization message) with a Hello adjacency previously created when Hellos were exchanged.

1. 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 Session Rejected/Parameters Error Notification message and closing the TCP connection.

   c. If LSR1 cannot find a matching Hello adjacency it sends a Session Rejected/No Hello Error Notification 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 an Error Notification message, LSR2 has rejected its proposed session and LSR1 closes the TCP connection.

2. When LSR1 plays the active role:

   a. If LSR1 receives an Error Notification message, LSR2 has rejected its proposed session 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 Session Rejected/Parameters Error Notification 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 with Error Notification 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.

The session establishment setup attempt following a NAK'd Initialization message must be delayed no less than 15 seconds, and subsequent delays must grow to a maximum delay of no less than 2 minutes. The specific session establishment action that must be delayed is the attempt to open the session transport connection by the LSR playing the active role.

The throttled sequence of Initialization NAKs is unlikely to cease until operator intervention reconfigures one of the LSRs. After such a configuration action there is no further need to throttle subsequent session establishment attempts (until their initialization messages are NAK'd).

Due to the asymmetric nature of session establishment, reconfiguration of the passive LSR will go unnoticed by the active LSR without some further action. Section "Hello Message" describes an optional mechanism an LSR can use to signal potential LDP peers that it has been reconfigured.

2.5.4. 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 (Active Role)	OPENSENT
	Receive acceptable Initialization msg (Passive Role) Action: Transmit Initialization msg and KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) and close transport connection	NON EXISTENT
OPENREC	Receive KeepAlive msg	OPERATIONAL
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) and close transport connection	NON EXISTENT
OPENSENT	Receive acceptable Initialization msg Action: Transmit KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) 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 Acceptable   | Tx Init msg         |
                      |        |    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 Acceptable        |
                      |        |                |    Init msg /        |
                      |        +----------------+ Tx KeepAlive msg     |
                      |                                                |
                      |      +-----------+                             |
                      +----->|           |                             |
                             |OPERATIONAL|                             |
                             |           |---------------------------->+
                             +-----------+     Rx Shutdown msg
                      All other  |   ^            or Timeout /
                        LDP msgs |   |         Tx Shutdown msg
                                 |   |
                                 +---+
```

2.5.5. 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 is 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 carry 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 that link (or target, in the case of Targeted Hellos) or that the peer has failed. The LSR then deletes the Hello adjacency. When the last Hello adjacency for a LDP session is deleted, the LSR terminates the LDP session by sending a Notification message and closing the transport connection.

2.5.6. Maintaining LDP Sessions

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

LDP uses the regular receipt of LDP PDUs on the session transport connection to monitor the integrity of the session. 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 LDP session by closing the transport connection.

After an LDP session has been established, an LSR must arrange that its peer receive 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.

2.6. Label Distribution and Management

The MPLS architecture [ARCH] allows an LSR to distribute a FEC label binding in response to an explicit request from another LSR. This is known as Downstream On Demand label distribution. It also allows an LSR to distribute label bindings to LSRs that have not explicitly requested them. [ARCH] calls this method of label distribution Unsolicited Downstream; this document uses the term Downstream Unsolicited.

Both of these label distribution techniques may be used in the same network at the same time. However, for any given LDP session, each LSR must be aware of the label distribution method used by its peer in order to avoid situations where one peer using Downstream Unsolicited label distribution assumes its peer is also. See Section "Downstream on Demand label Advertisement".

2.6.1. 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.

2.6.1.1. Independent Label Distribution Control

When using independent LSP control, each LSR 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 Unsolicited 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.

2.6.1.2. Ordered Label Distribution Control

When using LSP ordered control, an LSR may initiate the transmission of a label mapping only for a 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 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 boundary, such as another area for OSPF summary networks, or another autonomous system for OSPF AS externals and BGP routes [rfc1583] [rfc1771].

Note that whether an LSR is an egress for a given FEC may change over time, depending on the state of the network and LSR configuration settings.

2.6.2. Label Retention Mode

The MPLS architecture [ARCH] introduces the notion of label retention mode which specifies whether an LSR maintains a label binding for a FEC learned from a neighbor that is not its next hop for the FEC.

2.6.2.1. Conservative Label Retention Mode

In Downstream Unsolicited advertisement mode, label mapping advertisements for all routes may be received from all peer LSRs. When using conservative label retention, advertised label mappings are retained only 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, an LSR will request label mappings only from the 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 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.

2.6.2.2. Liberal Label Retention Mode

In Downstream Unsolicited advertisement mode, label mapping advertisements for all routes may be received from all LDP peers. When using liberal label retention, every label mappings received from a peer LSR is retained regardless of whether the LSR is the next hop for the advertised mapping. When operating in Downstream on Demand mode with liberal label retention, an LSR might choose to request label mappings for all known prefixes from all peer LSRs. Note, however, that Downstream on Demand mode is typically used by devices such as ATM switch-based LSRs for which 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.

2.6.3. Label Advertisement Mode

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

2.7. LDP Identifiers and Next Hop Addresses

An LSR maintains learned labels in a Label Information Base (LIB). When operating in Downstream Unsolicited mode, 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

2.8. Loop Detection

Loop detection is a configurable option which provides a mechanism for finding looping LSPs and for preventing Label Request messages from looping in the presence of non-merge capable LSRs.

The mechanism makes use of Path Vector and Hop Count TLVs carried by Label Request and Label Mapping messages. It builds on the following basic properties of these TLVs:

- A Path Vector TLV contains a list of the LSRs that its containing message has traversed. An LSR is identified in a Path Vector list by its unique LSR Identifier (Id), which is the first four octets of its LDP Identifier. When an LSR propagates a message containing a Path Vector TLV it adds its LSR Id to the Path Vector list. An LSR that receives a message with a Path Vector that contains its LSR Id detects that the message has traversed a loop. LDP supports the notion of a maximum allowable Path Vector length; an LSR that detects a Path Vector has reached the maximum length behaves as if the containing message has traversed a loop.

- A Hop Count TLV contains a count of the LSRS that the containing message has traversed. When an LSR propagates a message containing a Hop Count TLV it increments the count. An LSR that detects a Hop Count has reached a configured maximum value behaves as if the containing message has traversed a loop. By convention a count of 0 is interpreted to mean the hop count is unknown. Incrementing an unknown hop count value results in an unknown hop count value (0).

The following paragraphs describes LDP loop detection procedures. For these paragraphs, and only these paragraphs, "MUST" is redfined to mean "MUST if configured for loop detection". The paragraphs specify messages that must carry Path Vector and Hop Count TLVs. Note that the Hop Count TLV and its procedures are used without the Path Vector TLV in situations when loop detection is not configured (see [ATM] and [FR]).

2.8.1. Label Request Message

The use of the Path Vector TLV and Hop Count TLV prevent Label Request messages from looping in environments that include non-merge capable LSRs.

The rules that govern use of the Hop Count TLV in Label Request messages by LSR R when Loop Detection is enabled are the following:

- The Label Request message MUST include a Hop Count TLV.

- If R is sending the Label Request because it is a FEC ingress, it MUST include a Hop Count TLV with hop count value 1.

- If R is sending the Label Request as a result of having received a Label Request from an upstream LSR, and if the received Label Request contains a Hop Count TLV, R MUST increment the received hop count value by 1 and MUST pass the resulting value in a Hop Count TLV to its next hop along with the Label Request message;

The rules that govern use of the Path Vector TLV in Label Request messages by LSR R when Loop Detection is enabled are the following:

- If R is sending the Label Request because it is a FEC ingress, then if R is non-merge capable, it MUST include a Path Vector TLV of length 1 containing its own LSR Id.

- If R is sending the Label Request as a result of having received a Label Request from an upstream LSR, then if the received Label Request contains a Path Vector TLV or if R is non-merge capable:

    R MUST add its own LSR Id to the Path Vector, and MUST pass the resulting Path Vector to its next hop along with the Label Request message. If the Label Request contains no Path Vector TLV, R MUST include a Path Vector TLV of length 1 containing its own LSR Id.

Note that if R receives a Label Request message for a particular FEC, and R has previously sent a Label Request message for that FEC to its next hop and has not yet received a reply, and if R intends to merge the newly received Label Request with the existing outstanding Label Request, then R does not propagate the Label Request to the next hop.

If R receives a Label Request message from its next hop with a Hop Count TLV which exceeds the configured maximum value, or with a Path Vector TLV containing its own LSR Id or which exceeds the maximum allowable length, then R detects that the Label Request message has traveled in a loop.

When R detects a loop, it MUST send a Loop Detected Notification message to the source of the Label Request message and drop the Label Request message.

2.8.2. Label Mapping Message

The use of the Path Vector TLV and Hop Count TLV in the Label Mapping message provide a mechanism to find and terminate looping LSPs. When an LSR receives a Label Mapping message from a next hop, the message is propagated upstream as specified below until an ingress LSR is reached or a loop is found.

The rules that govern the use of the Hop Count TLV in Label Mapping messages sent by an LSR R when Loop Detection is enabled are the following:

- R MUST include a Hop Count TLV.

- If R is the egress, the hop count value MUST be 1.

- If the Label Mapping message is being sent to propagate a Label Mapping message received from the next hop to an upstream peer, the hop count value MUST be determined as follows:

  o If R is a member of the edge set of an LSR domain whose LSRs do not perform 'TTL-decrement' (e.g., an ATM LSR domain or a Frame Relay LSR domain) and the upstream peer is within that domain, R MUST reset the hop count to 1 before propagating the message.

  o Otherwise, R MUST increment the hop count received from the next hop before propagating the message.

- If the Label Mapping message is not being sent to propagate a Label Mapping message, the hop count value MUST be the result of incrementing R's current knowledge of the hop count learned from previous Label Mapping messages. Note that this hop count value will be unknown if R has not received a Label Mapping message from the next hop.

Any Label Mapping message MAY contain a Path Vector TLV. The rules that govern the mandatory use of the Path Vector TLV in Label Mapping messages sent by LSR R when Loop Detection is enabled are the following:

- If R is the egress, the Label Mapping message need not include a Path Vector TLV.

- If R is sending the Label Mapping message to propagate a Label Mapping message received from the next hop to an upstream peer, then:

    o If R is merge capable and if R has not previously sent a Label Mapping message to the upstream peer, then it MUST include a Path Vector TLV.

    o If the received message contains an unknown hop count, then R MUST include a Path Vector TLV.

    o If R has previously sent a Label Mapping message to the upstream peer, then it MUST include a Path Vector TLV if the received message reports an LSP hop count increase, a change in hop count from unknown to known, or a change from known to unknown.

  If the above rules require R include a Path Vector TLV in the Label Mapping message, R computes it as follows:

    o If the received Label Mapping message included a Path Vector, the Path Vector sent upstream MUST be the result of adding R's LSR Id to the received Path Vector.

    o If the received message had no Path Vector, the Path Vector sent upstream MUST be a path vector of length 1 containing R's LSR Id.

- If the Label Mapping message is not being sent to propagate a received message upstream, the Label Mapping message MUST include a Path Vector of length 1 containing R's LSR Id.

If R receives a Label Mapping message from its next hop with a Hop Count TLV which exceeds the configured maximum value, or with a Path Vector TLV containing its own LSR Id or which exceeds the maximum allowable length, then R detects that the corresponding LSP contains a loop.

When R detects a loop, it MUST stop using the label for forwarding, drop the Label Mapping message, and signal Loop Detected status to the source of the Label Mapping message.

2.8.3. Discussion

If loop detection is desired in an MPLS domain, then it should be turned on in ALL LSRs within that MPLS domain, else loop detection will not operate properly and may result in undetected loops or in falsely detected loops.

LSRs which are configured for loop detection are NOT expected to store the path vectors as part of the LSP state.

Note that in a network where only non-merge capable LSRs are present, Path Vectors are passed downstream from ingress to egress, and are not passed upstream. Even when merge is supported, Path Vectors need not be passed upstream along an LSP which is known to reach the egress. When an LSR experiences a change of next hop, it need pass Path Vectors upstream only when it cannot tell from the hop count that the change of next hop does not result in a loop.

In the case of ordered label distribution, Label Mapping messages are propagated from egress toward ingress, naturally creating the Path Vector along the way. In the case of independent label distribution, an LSR may originate a Label Mapping message for an FEC before receiving a Label Mapping message from its downstream peer for that FEC. In this case, the subsequent Label Mapping message for the FEC received from the downstream peer is treated as an update to LSP attributes, and the Label Mapping message must be propagated upstream. Thus, it is recommended that loop detection be configured in conjunction with ordered label distribution, to minimize the number of Label Mapping update messages.

2.9. Authenticity and Integrity of LDP Messages

This section specifies a mechanism to protect against the introduction of spoofed TCP segments into LDP session connection streams. The use of this mechanism MUST be supported as a configurable option.

The mechanism is based on use of the TCP MD5 Signature Option specified in [rfc2385] for use by BGP. See [rfc1321] for a specification of the MD5 hash function.

2.9.1. TCP MD5 Signature Option

The following quotes from [rfc2385] outline the security properties achieved by using the TCP MD5 Signature Option and summarizes its operation:

"IESG Note

This document describes current existing practice for securing BGP against certain simple attacks. It is understood to have security weaknesses against concerted attacks."

"Abstract

This memo describes a TCP extension to enhance security for BGP. It defines a new TCP option for carrying an MD5 [RFC1321] digest in a TCP segment. This digest acts like a signature for that segment, incorporating information known only to the connection end points. Since BGP uses TCP as its transport, using this option in the way described in this paper significantly reduces the danger from certain security attacks on BGP."

"Introduction

The primary motivation for this option is to allow BGP to protect itself against the introduction of spoofed TCP segments into the connection stream. Of particular concern are TCP resets.

To spoof a connection using the scheme described in this paper, an attacker would not only have to guess TCP sequence numbers, but would also have had to obtain the password included in the MD5 digest. This password never appears in the connection stream, and the actual form of the password is up to the application. It could even change during the lifetime of a particular connection so long as this change was synchronized on both ends (although retransmission can become problematical in some TCP implementations with changing passwords).

Finally, there is no negotiation for the use of this option in a connection, rather it is purely a matter of site policy whether or not its connections use the option."

"MD5 as a Hashing Algorithm

Since this memo was first issued (under a different title), the MD5 algorithm has been found to be vulnerable to collision

search attacks [Dobb], and is considered by some to be insufficiently strong for this type of application.

This memo still specifies the MD5 algorithm, however, since the option has already been deployed operationally, and there was no "algorithm type" field defined to allow an upgrade using the same option number. The original document did not specify a type field since this would require at least one more byte, and it was felt at the time that taking 19 bytes for the complete option (which would probably be padded to 20 bytes in TCP implementations) would be too much of a waste of the already limited option space.

This does not prevent the deployment of another similar option which uses another hashing algorithm (like SHA-1). Also, if most implementations pad the 18 byte option as defined to 20 bytes anyway, it would be just as well to define a new option which contains an algorithm type field.

This would need to be addressed in another document, however."

End of quotes from [rfc2385].

2.9.2. LDP Use of TCP MD5 Signature Option

LDP uses the TCP MD5 Signature Option as follows:

- Use of the MD5 Signature Option for LDP TCP connections is a configurable LSR option.

- An LSR that uses the MD5 Signature Option is configured with a password (shared secret) for each potential LDP peer.

- The LSR applies the MD5 algorithm as specified in [RFC2385] to compute the MD5 digest for a TCP segment to be sent to a peer. This computation makes use of the peer password as well as the TCP segment.

- When the LSR receives a TCP segment with an MD5 digest, it validates the segment by calculating the MD5 digest (using its own record of the password) and compares the computed digest with the received digest. If the comparison fails, the segment is dropped without any response to the sender.

   - The LSR ignores LDP Hellos from any LSR for which a password has
     not been configured. This ensures that the LSR establishes LDP
     TCP connections only with LSRs for which a password has been
     configured.

2.10. Label Distribution for Explicitly Routed LSPs

Traffic Engineering [TE] is expected to be an important MPLS application. MPLS support for Traffic Engineering uses explicitly routed LSPs, which need not follow normally-routed (hop-by-hop) paths as determined by destination-based routing protocols. CR-LDP [CRLDP] defines extensions to LDP to use LDP to set up explicitly routed LSPs.

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 signaling some event.

3.1. LDP PDUs

Each LDP PDU is an LDP header followed by one or more LDP messages. The 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                        |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 octets, excluding the Version and PDU Length fields.

  The maximum allowable PDU Length is negotiable when an LDP session is initialized. Prior to completion of the negotiation the maximum allowable length is 4096 bytes.

LDP Identifier
  Six octet field that uniquely identifies the label space of the sending LSR for which this PDU applies. The first four octets identify the LSR and must be a globally unique value. It should be a 32-bit router Id assigned to the LSR and also used to identify it in loop detection Path Vectors. The last two octets identify a label space within the LSR. For a platform-wide label space, these should both be zero.

Note that there is no alignment requirement for the first octet of an LDP PDU.

3.2. LDP Procedures

LDP defines messages, TLVs and procedures in the following areas:

- Peer discovery;
- Session management;
- Label distribution;
- Notification of errors and advisory information.

The sections that follow describe the message and TLV encodings for these areas and the procedures that apply to them.

The label distribution procedures are complex and are difficult to describe fully, coherently and unambiguously as a collection of separate message and TLV specifications.

Appendix A, "LDP Label Distribution Procedures", describes the label distribution procedures in terms of label distribution events that may occur at an LSR and how the LSR must respond. Appendix A is the specification of LDP label distribution procedures. If a procedure described elsewhere in this document conflicts with Appendix A, Appendix A specifies LDP behavior.

3.3. Type-Length-Value Encoding

LDP uses a Type-Length-Value (TLV) encoding scheme to encode much of the information carried in LDP messages.

An LDP TLV is encoded as a 2 octet field that uses 14 bits to specify a Type and 2 bits to specify behavior when an LSR doesn't recognize the Type, 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|        Type               |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                             Value                             |
~                                                               ~
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

U bit
  Unknown TLV bit. Upon receipt of an unknown TLV, if U is clear (=0), a notification must be returned to the message originator and the entire message must be ignored; if U is set (=1), the unknown TLV is silently ignored and the rest of the message is processed as if the unknown TLV did not exist. The sections following that define TLVs specify a value for the U-bit.

F bit
  Forward unknown TLV bit. This bit applies only when the U bit is set and the LDP message containing the unknown TLV is to be forwarded. If F is clear (=0), the unknown TLV is not forwarded with the containing message; if F is set (=1), the unknown TLV is forwarded with the containing message. The sections following that define TLVs specify a value for the F-bit.

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 to be

interpreted as specified by the Type field.

Note that there is no alignment requirement for the first octet of a TLV.

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 specification 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 it 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 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 section in this document that describes each.

3.4. TLV Encodings for Commonly Used Parameters

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

3.4.1. FEC TLV

Labels are bound to Forwarding Equivalence Classes (FECs). a 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0|0| 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 "FECs". The
  FEC element encoding depends on the type of FEC element.

  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. Note that while the representation of the
  FEC element value is type-dependent, the FEC element encoding
  itself is one where standard LDP TLV encoding is not used.

  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 below.
Host Address	0x03	Full host address; see below.

  Note that this version of LDP supports the use of multiple FEC
  Elements per FEC for the Label Mapping message only. The use of
  multiple FEC Elements in other messages is not permitted in this
  version, and is a subject for future study.

  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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Prefix (2)   |     Address Family            |     PreLen    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     Prefix                                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Family
  Two octet quantity containing a value from ADDRESS FAMILY
  NUMBERS in [rfc1700] that encodes the address family for the
  address prefix in the Prefix field.

PreLen
  One octet unsigned integer containing the length in bits of the
  address prefix that follows. A length of zero indicates a
  prefix that matches all addresses (the default destination); in
  this case the Prefix itself is zero octets).

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.

Host Address FEC Element 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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Host Addr (3) |     Address Family            | Host Addr Len |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     |                     Host Addr                                 |
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Family
  Two octet quantity containing a value from ADDRESS FAMILY
  NUMBERS in [rfc1700] that encodes the address family for the
  address prefix in the Prefix field.

Host Addr Len
  Length of the Host address in octets.

Host Addr

An address encoded according to the Address Family field.

3.4.1.1. FEC Procedures

If in decoding a FEC TLV an LSR encounters a FEC Element with an Address Family it does not support, it should stop decoding the FEC TLV, abort processing the message containing the TLV, and send an "Unsupported Address Family" Notification message to its LDP peer signaling an error.

If it encounters a FEC Element type it cannot decode, it should stop decoding the FEC TLV, abort processing the message containing the TLV, and send an "Unknown FEC" Notification message to its LDP peer signaling an error.

3.4.2. 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.

3.4.2.1. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| 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.

3.4.2.2. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| 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
  significant. 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 Channel 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.

3.4.2.3. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| 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	
2 = 23 bits DLCI	

  Len values 1 and 3 are reserved.

DLCI
  The Data Link Connection Identifier. Refer to [FR] for the label
  values and formats.

3.4.3. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| Address List (0x0101)     |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Address Family            |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                        Addresses                              |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Family
  Two octet quantity containing a value from ADDRESS FAMILY NUMBERS
  in [rfc1700] 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

3.4.4. 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.

Note that setup procedures for LSPs that traverse ATM and Frame Relay links require use of the Hop Count TLV (see [ATM] and [FR]).

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

HC Value
  1 octet unsigned integer hop count value.

3.4.4.1. Hop Count Procedures

During setup of an LSP an LSR R 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 LSR R then propagates the Label Mapping message for the LSP to an upstream peer or the Label Request message to a downstream peer to continue the LSP setup, it must must determine a hop count to include in the propagated message as follows:

- If the message is a Label Request message, R must increment the received hop count;

- If the message is a Label Mapping message, R determines the hop count as follows:

- o If R is a member of the edge set of an LSR domain whose LSRs do not perform 'TTL-decrement' and the upstream peer is within that domain, R must reset the hop count to 1 before propagating the message.

- o Otherwise, R must increment the received hop count.

The first LSR in the LSP (ingress for a Label Request message, egress for a Label Mapping message) should set the hop count value to 1.

By convention a value of 0 indicates an unknown hop count. The result of incrementing an unknown hop count is itself an unknown hop count (0).

Use of the unknown hop count value greatly reduces the signaling overhead when independent control is used. When a new LSP is established, each LSR starts with unknown hop count. Addition of a new LSR whose hop count is also unknown does not cause a hop count update to be propagated upstream since the hop count remains unknown. When the egress is finally added to the LSP, then the LSRs propagate hop count updates upstream via Label Mapping messages.

Without use of the unknown hop count, each time a new LSR is added to the LSP a hop count update would need to be propagated upstream if the new LSR is closer to the egress than any of the other LSRs. These updates are useless overhead since they don't reflect the hop count to the egress.

From the perspective of the ingress node, the fact that the hop count is unknown implies nothing about whether a packet sent on the LSP will actually make it to the egress. All it implies is that the hop count update from the egress has not yet reached the ingress.

If an LSR receives a message containing a Hop Count TLV, it must check the hop count value to determine whether the hop count has exceeded its configured maximum allowable value. If so, it must behave as if the containing message has traversed a loop by sending a Notification message signaling Loop Detected in reply to the sender of the message.

If Loop Detection is configured, the LSR must follow the procedures specified in Section "Loop Detection".

3.4.5. Path Vector TLV

The Path Vector TLV is used with the Hop Count TLV in Label Request and Label Mapping messages to implement the optional LDP loop detection mechanism. See Section "Loop Detection". Its use in the Label Request message records the path of LSRs the request has traversed. Its use in the Label Mapping message 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| Path Vector (0x0104)      |        Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            LSR Id 1                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            LSR Id n                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

One or more LSR Ids
  A list of router-ids indicating the path of LSRs the message has traversed. Each LSR Id is the first four octets (router-id) of the LDP identifier for the corresponding LSR. This ensures it is unique within the LSR network.

3.4.5.1. Path Vector Procedures

The Path Vector TLV is carried in Label Mapping and Label Request messages when loop detection is configured.

3.4.5.1.1. Label Request Path Vector

Section "Loop Detection" specifies situations when an LSR must include a Path Vector TLV in a Label Request message.

An LSR that receives a Path Vector in a Label Request message must perform the procedures described in Section "Loop Detection".

If the LSR detects a loop, it must reject the Label Request message.

The LSR must:

1. Transmit a Notification message to the sending LSR signaling "Loop Detected".

2. Not propagate the Label Request message further.

Note that a Label Request message with Path Vector TLV is forwarded until:

1. A loop is found,

2. The LSP egress is reached,

3. The maximum Path Vector limit or maximum Hop Count limit is reached. This is treated as if a loop had been detected.

3.4.5.1.2. Label Mapping Path Vector

Section "Loop Detection" specifies the situations when an LSR must include a Path Vector TLV in a Label Mapping message.

An LSR that receives a Path Vector in a Label Mapping message must perform the procedures described in Section "Loop Detection".

If the LSR detects a loop, it must reject the Label Mapping message in order to prevent a forwarding loop. The LSR must:

1. Transmit a Label Release message carrying a Status TLV to the sending LSR to signal "Loop Detected".

2. Not propagate the message further.

3. Check whether the Label Mapping message is for an existing LSP. If so, the LSR must unsplice any upstream labels which are spliced to the downstream label for the FEC.

Note that a Label Mapping message with a Path Vector TLV is forwarded until:

1. A loop is found,

2. An LSP ingress is reached, or

      3. The maximum Path Vector or maximum Hop Count limit is reached.
         This is treated as if a loop had been detected.

3.4.6. Status TLV

Notification messages carry Status TLVs to specify events being signaled.

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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |U|F| Status (0x0300)           |      Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Status Code                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Message Type             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

U bit
  Should be 0 when the Status TLV is sent in a Notification message.
  Should be 1 when the Status TLV is sent in some other message.

F bit
  Should be the same as the setting of the F-bit in the Status Code field.

Status Code
  32-bit unsigned integer encoding the event being signaled. 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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |E|F|                 Status Data                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

  F bit
    Forward 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 signaled.  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
     message type.
```

Note that use of the Status TLV is not limited to Notification messages. A message other than a Notification message may carry a Status TLV as an Optional Parameter. When a message other than a Notification carries a Status TLV the U-bit of the Status TLV should be set to 1 to indicate that the receiver should silently discard the TLV if unprepared to handle it.

3.5. LDP Messages

All LDP messages have the following format:

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

U bit
  Unknown message bit. Upon receipt of an unknown message, if U is
  clear (=0), a notification is returned to the message originator;
  if U is set (=1), the unknown message is silently ignored. The
  sections following that define messages specify a value for the U-
  bit.

Message Type
  Identifies the type of message

Message Length
  Specifies the cumulative length in octets of the Message ID,
  Mandatory Parameters, and Optional Parameters.

Message ID
  32-bit value 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 should include this Message Id in the Status TLV
  carried by 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.

Note that there is no alignment requirement for the first octet of an LDP message.

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

Message Name	Section Title
Notification	"Notification Message"
Hello	"Hello Message"
Initialization	"Initialization Message"
KeepAlive	"KeepAlive Message"
Address	"Address Message"
Address Withdraw	"Address Withdraw Message"
Label Mapping	"Label Mapping Message"
Label Request	"Label Request Message"
Label Abort Request	"Label Abort Request Message"
Label Withdraw	"Label Withdraw Message"
Label Release	"Label Release Message"

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

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

While it 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.

3.5.1. 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 such as the outcome of processing an LDP message 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0|   Notification (0x0001)     |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Status (TLV)                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

Status TLV
  Indicates the event being signaled. 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
Returned PDU	0x0302	var	See below
Returned Message	0x0303	var	See below

Other Optional Parameters, specific to the particular event being signaled 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.

Returned PDU
An LSR uses this parameter to return part of an LDP PDU to the LSR that sent it. The value of this TLV is the PDU header and as much PDU data following the header as appropriate for the condition being signaled by the Notification message.

Returned Message
An LSR uses this parameter to return part of an LDP message to the LSR that sent it. The value of this TLV is the message type and length fields and as much message data following the type and length fields as appropriate for the condition being signaled by the Notification message.

3.5.1.1. 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 condition.

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.

3.5.1.2. Events Signaled by Notification Messages

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

3.5.1.2.1. 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 peer for this LDP session. This is a fatal error signaled 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 signaled by the Bad Protocol Version Status Code.

- The PDU Length field is too small (< 14) or too large (> maximum PDU length). This is a fatal error signaled by the Bad PDU Length Status Code. Section "Initialization Message" describes how the maximum PDU length for a session is determined.

An LDP Message is malformed if:

- The Message Type is unknown.

  If the Message Type is < 0x8000 (high order bit = 0) it is an error signaled by the Unknown Message Type Status Code.

  If the Message Type is >= 0x8000 (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 signaled by the Bad Message Length Status Code.

- The message is missing one or more Mandatory Parameters. This is a non-fatal error signalled by the Missing Message Parameters Status Code.

3.5.1.2.2. 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 signaled by the Bad TLV Length Status Code.

- The TLV type is unknown.

  If the TLV type is < 0x8000 (high order bit 0) it is an error signaled by the Unknown TLV Status Code.

  If the TLV type is >= 0x8000 (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 interpreted as indicative of a bug in either the sending or receiving LSR. It is a fatal error signaled by the Malformed TLV Value Status Code.

3.5.1.2.3. Session KeepAlive Timer Expiration

This is a fatal error signaled by the KeepAlive Timer Expired Status Code.

3.5.1.2.4. Unilateral Session Shutdown

This is a fatal event signaled by the Shutdown Status Code. The Notification Message may optionally include an Extended Status TLV to provide a reason for the Shutdown. The sending LSR terminates the session immediately after sending the Notification.

3.5.1.2.5. 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 is defined in Sections "Initialization Message".

3.5.1.2.6. Events Resulting From Other Messages

Messages other than the Initialization message may result in events that must be signaled 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.

3.5.1.2.7. Internal Errors

An LDP implementation may be capable of detecting problem conditions specific to its implementation. When such a condition prevents an implementation from interacting correctly with a peer, the implementation should, when capable of doing so, use the Internal Error Status Code to signal the peer. This is a fatal error.

3.5.1.2.8. 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.

3.5.2. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Hello (0x0100)            |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Common Hello Parameters TLV               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

Common Hello Parameters TLV
  Specifies parameters common to all Hello messages. The encoding
  for the Common Hello 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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |0|0| Common Hello Parms(0x0400)|      Length                   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      Hold Time                |T|R| Reserved                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Hold Time,
Hello hold time in seconds. An LSR maintains a record of Hellos received from potential peers (see Section "Hello Message Procedures"). Hello Hold Time specifies the time the sending LSR will maintain its record of Hellos from the receiving LSR without receipt of another Hello.

A pair of LSRs negotiates the hold times they use for Hellos from each other. Each proposes a hold time. The hold time used is the minimum of the hold times proposed in their Hellos.

A value of 0 means use the default, which is 15 seconds for Link Hellos and 45 seconds for Targeted Hellos. A value of 0xffff means infinite.

T, Targeted Hello
A value of 1 specifies that this Hello is a Targeted Hello. A value of 0 specifies that this Hello is a Link Hello.

R, Request Send Targeted Hellos
A value of 1 requests the receiver to send periodic Targeted Hellos to the source of this Hello. A value of 0 makes no request.

An LSR initiating Extended Discovery sets R to 1. If R is 1, the receiving LSR checks whether it has been configured to send Targeted Hellos to the Hello source in response to Hellos with this request. If not, it ignores the request. If so, it initiates periodic transmission of Targeted Hellos to the Hello source.

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

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
IPv4 Transport Address	0x0401	4	See below
Configuration Sequence Number	0x0402	4	See below
IPv6 Transport Address	0x0403	8	See below

IPv4 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.

Configuration Sequence Number
Specifies a 4 octet unsigned configuration sequence number that identifies the configuration state of the sending LSR. Used by the receiving LSR to detect configuration changes on the sending LSR.

IPv6 Transport Address
Specifies the IPv6 address to be used for the sending LSR when opening the LDP session TCP connection. If this optional TLV is not present the IPv6 source address for the UDP packet carrying the Hello should be used.

3.5.2.1. Hello Message Procedures

An LSR receiving Hellos from another LSR maintains a Hello adjacency corresponding to 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".

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

An 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 PDU header for the Hello, it follows the procedures of 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 source address A 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.

The following describes how an LSR processes Hello optional TLVs:

Transport Address
The LSR associates the specified transport address with the Hello adjacency.

Configuration Sequence Number
The Configuration Sequence Number optional parameter is used by the sending LSR to signal configuration changes to the receiving LSR. When a receiving LSR playing the active role in LDP session establishment detects a change in the sending LSR configuration, it may clear the session setup backoff delay, if any, associated with the sending LSR (see Section "Session Initialization").

A sending LSR using this optional parameter is responsible for maintaining the configuration sequence number it transmits in Hello messages. Whenever there is a configuration change on the sending LSR, it increments the configuration sequence number.

3.5.3. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Initialization (0x0200)   |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Common Session Parameters TLV             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

Common Session Parameters TLV
  Specifies values proposed by the sending LSR for parameters that must be negotiated for every LDP session.

  The encoding for the Common 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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|0| Common Sess Parms (0x0500)|      Length                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Protocol Version              |      KeepAlive Time           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |A|D|  Reserved |     PVLim     |      Max PDU Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                 Receiver LDP Identifier                       |
    +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |
    -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-++
```

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

protocol version 1.

KeepAlive Time

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

A, Label Advertisement Discipline

Indicates the type of Label advertisement. A value of 0 means Downstream Unsolicited advertisement; a value of 1 means Downstream On Demand.

If one LSR proposes Downstream Unsolicited and the other proposes Downstream on Demand, the rules for resolving this difference is:

- If the session is for a label-controlled ATM link or a label-controlled Frame Relay link, then Downstream on Demand must be used.

- Otherwise, Downstream Unsolicited must be used.

If the label advertisement discipline determined in this way is unacceptable to an LSR, it must send a Session Rejected/Parameters Advertisement Mode Notification message in response to the Initialization message and not establish the session.

D, Loop Detection

Indicates whether loop detection based on path vectors is enabled. A value of 0 means loop detection is disabled; a value of 1 means that loop detection is enabled.

PVLim, Path Vector Limit

The configured maximum path vector length. Must be 0 if loop detection is disabled (D = 0). If the loop detection procedures would require the LSR to send a path vector that exceeds this limit, the LSR will behave as if a loop had been detected for the FEC in question.

When Loop Detection is enabled in a portion of a network, it is recommended that all LSRs in that portion of the network be

configured with the same path vector limit. Although knowledge of a peer's path vector limit will not change an LSR's behavior, it does enable the LSR to alert an operator to a possible misconfiguration.

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

Max PDU Length
Two octet unsigned integer that proposes the maximum allowable length for LDP PDUs for the session. A value of 255 or less specifies the default maximum length of 4096 octets.

The receiving LSR MUST calculate the maximum PDU length for the session by using the smaller of its and its peer's proposals for Max PDU Length. The default maximum PDU length applies before session initialization completes.

If the maximum PDU length determined this way is unacceptable to an LSR, it must send a Session Rejected/Parameters Max PDU Length Notification message in response to the Initialization message and not establish the session.

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

If there is no matching Hello adjacency, the LSR must send a Session Rejected/No Hello Notification message in response to the Initialization message and not establish the session.

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
ATM Session Parameters	0x0501	var	See below
Frame Relay Session Parameters	0x0502	var	See below

ATM Session Parameters
Used when an LDP session manages label exchange for an ATM link

to specify ATM-specific session parameters.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0|   ATM Sess Parms (0x0501) |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| M |   N   |D|                        Reserved                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ATM Label Range Component 1                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ATM Label Range Component N                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M, ATM Merge Capabilities
  Specifies the merge capabilities of an ATM switch. The following values are supported in this version of the specification:

Value	Meaning
0	Merge not supported
1	VP Merge supported
2	VC Merge supported
3	VP & VC Merge supported

If the merge capabilities of the LSRs differ, then:

- Non-merge and VC-merge LSRs may freely interoperate.

- The interoperability of VP-merge-capable switches with non-VP-merge-capable switches is a subject for future study. When the LSRs differ on the use of VP-merge, the session is established, but VP merge is not used.

Note that if VP merge is used, it is the responsibility of the ingress node to ensure that the chosen VCI is unique within the LSR domain (see [ATM-VP]).

N, Number of label range components
  Specifies the number of ATM Label Range Components included in the TLV.

D, VC Directionality
  A value of 0 specifies bidirectional VC capability, meaning the LSR can (within a given VPI) support the use of a given VCI as a label for both link directions independently. A value of 1 specifies unidirectional VC capability, meaning (within a given VPI) a given VCI may appear in a label mapping for one direction on the link only. When either or both of the peers specifies unidirectional VC capability, both LSRs use unidirectional VC label assignment for the link as follows. The LSRs compare their LDP Identifiers as unsigned integers. The LSR with the larger LDP Identifier may assign only odd-numbered VCIs in the VPI/VCI range as labels. The system with the smaller LDP Identifier may assign only even-numbered VCIs in the VPI/VCI range as labels.

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

One or more ATM Label Range Components
  A list of ATM Label Range Components which together specify 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 MUST NOT establish a session with neighbors for which the intersection of ranges is NULL. In this case, the LSR must send a Session Rejected/Parameters Label Range Notification message in response to the Initialization message and not establish the session.

  The encoding for an ATM Label Range Component 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  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.

When peer LSRs are connected indirectly by means of an ATM VP, the sending LSR should set the Minimum and Maximum VPI fields to 0, and the receiving LSR must ignore the Minimum and Maximum VPI fields.

See [ATM-VP] for specification of the fields for ATM Label Range Components to be used with VP merge LSRs.

Frame Relay Session Parameters
Used when an LDP session manages label exchange for a Frame Relay link to specify Frame Relay-specific session parameters.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |0|0|   FR Sess Parms (0x0502)  |      Length                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | M |   N    |D|                       Reserved                 |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             Frame Relay Label Range Component 1               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       ~                                                               ~
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             Frame Relay Label Range Component N               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M, Frame Relay Merge Capabilities
  Specifies the merge capabilities of a Frame Relay switch. The following values are supported in this version of the specification:

Value	Meaning
0	Merge not supported
1	Merge supported

  Non-merge and merge Frame Relay LSRs may freely interoperate.

N, Number of label range components
  Specifies the number of Frame Relay Label Range Components included in the TLV.

D, VC Directionality
  A value of 0 specifies bidirectional VC capability, meaning the LSR can support the use of a given DLCI as a label for both link directions independently. A value of 1 specifies unidirectional VC capability, meaning a given DLCI may appear in a label mapping for one direction on the link only. When either or both of the peers specifies unidirectional VC capability, both LSRs use unidirectional VC label assignment for the link as follows. The LSRs compare their LDP Identifiers as unsigned integers. The LSR with the larger LDP Identifier may assign only odd-numbered DLCIs in the range as labels. The system with the smaller LDP Identifier may assign only even-numbered DLCIs in the range as labels.

Reserved

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

One or more Frame Relay Label Range Components

A list of Frame Relay Label Range Components which together specify 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 MUST NOT establish a session with neighbors for which the intersection of ranges is NULL. In this case, the LSR must send a Session Rejected/Parameters Label Range Notification message in response to the Initialization message and not establish the session.

The encoding for a Frame Relay Label Range Component 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved    |Len|                     Minimum DLCI            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved        |                     Maximum DLCI            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Reserved

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

Len

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

Len	DLCI bits
0	10
2	23

Len values 1 and 3 are reserved.

Minimum DLCI

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

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

Note that there is no Generic Session Parameters TLV for sessions which advertise Generic Labels.

3.5.3.1. Initialization Message Procedures

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

3.5.4. 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0|   KeepAlive (0x0201)        |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Message ID
     32-bit value used to identify this message.

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

3.5.4.1. KeepAlive Message Procedures

The KeepAlive Timer mechanism described in Section "Maintaining LDP Sessions" resets a session KeepAlive timer every time an LDP PDU is received on the session TCP connection. The KeepAlive Message is provided to allow reset of the KeepAlive Timer in circumstances where

an LSR has no other information to communicate to an LDP peer.

An LSR must arrange that its peer receive an LDP Message from it at least every KeepAlive Time period. Any LDP protocol message will do but, in circumstances where no other LDP protocol messages have been sent within the period, a KeepAlive message must be sent.

3.5.5. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Address (0x0300)          |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Address List TLV                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value 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.

3.5.5.1. 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 an 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".

If an LSR does not support the Address Family specified in the Address List TLV, it should send an "Unsupported Address Family" Notification to its LDP signalling an error and abort processing the message.

3.5.6. Address Withdraw Message

An LSR sends the Address Withdraw 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Address Withdraw (0x0301) |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Address List TLV                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value 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.

3.5.6.1. Address Withdraw Message Procedures

See Section "Address Message Procedures"

3.5.7. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Label Mapping (0x0400)    |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC TLV                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Label TLV                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

FEC TLV
  Specifies the FEC component of the FEC-Label mapping being advertised. See Section "FEC TLV" for encoding.

Label TLV
  Specifies the Label component of the FEC-Label mapping. See Section "Label TLV" for encoding.

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

Optional Parameter	Length	Value

```
         Label Request         4            See below
             Message ID TLV
         Hop Count TLV         1            See below
         Path Vector TLV       variable     See below
```

The encodings for the Hop Count, and Path Vector TLVs can be found in Section "TLV Encodings for Commonly Used Parameters".

Label Request Message ID
If this Label Mapping message is a response to a Label Request message it must include the Request Message Id optional parameter. The value of this optional parameter is the Message Id of the corresponding Label Request Message.

Hop Count
Specifies the running total of the number of LSR hops along the LSP being setup by the Label Message. Section "Hop Count Procedures" describes how to handle this TLV.

Path Vector
Specifies the LSRs along the LSP being setup by the Label Message. Section "Path Vector Procedures" describes how to handle this TLV.

3.5.7.1. Label Mapping Message Procedures

The Mapping message is used by an LSR to distribute a label mapping for a FEC to an LDP peer. 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 responsible for the consistency of the label mappings it has distributed, and that its peers have these mappings.

An LSR receiving a Label Mapping message from a downstream LSR for a Prefix or Host Address FEC Element should not use the label for forwarding unless its routing table contains an entry that exactly matches the FEC Element.

See Appendix A "LDP Label Distribution Procedures" for more details.

3.5.7.1.1. Independent Control Mapping

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

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

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

3. The next hop for a 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.

3.5.7.1.2. 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 a 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 a 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.

3.5.7.1.3. 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 Unsolicited advertisement 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 from Rd to Ru.

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.

3.5.7.1.4. Downstream Unsolicited 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.

The combination of Downstream Unsolicited mode and conservative label retention can lead to a situation where an LSR releases the label for a FEC that it later needs. For example, if LSR Rd advertises to LSR Ru the label for a FEC for which it is not Ru's next hop, Ru will release the label. If Ru's next hop for the FEC later changes to Rd, it needs the previously released label.

To deal with this situation either Ru can explicitly request the label when it needs it, or Rd can periodically readvertise it to Ru. In many situations Ru will know when it needs the label from Rd. For example, when its next hop for the FEC changes to Rd. However, there could be situations when Ru does not. For example, Rd may be attempting to establish an LSP with non-standard properties. Forcing Ru to explicitly request the label in this situation would require it to maintain state about a potential LSP with non-standard properties.

In situations where Ru knows it needs the label, it is responsible for explicitly requesting the label by means of a Label Request message. In situations where Ru may not know that it needs the label, Rd is responsible for periodically readvertising the label to Ru.

For this version of LDP, the only situation where Ru knows it needs a label for a FEC from Rd is when Rd is its next hop for the FEC, Ru does not have a label from Rd, and the LSP for the FEC is one that can be established with TLVs defined in this document.

3.5.8. Label Request Message

An LSR sends the Label Request Message to an LDP peer to request a binding (mapping) for a FEC.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Label Request (0x0401)    |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC TLV                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

FEC TLV
  The FEC for which a label is being requested. See Section "FEC TLV" for encoding.

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

Optional Parameter	Length	Value
Hop Count TLV	1	See below
Path Vector TLV	variable	See below

  The encodings for the Hop Count, and Path Vector TLVs can be found in Section "TLV Encodings for Commonly Used Parameters".

    Hop Count
      Specifies the running total of the number of LSR hops along the LSP being setup by the Label Request Message. Section "Hop Count Procedures" describes how to handle this TLV.

    Path Vector
      Specifies the LSRs along the LSR being setup by the Label Request Message. Section "Path Vector Procedures" describes how to handle this TLV.

3.5.8.1. 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 a FEC.

An LSR may transmit a Request message under any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table, and the next hop is an 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.

   Note that if the LSR already has a pending Label Request message for the new next hop it should not issue an additional Label Request in response to the next hop change.

3. The LSR receives a Label Request for a FEC from an upstream LDP peer, the FEC next hop is an LDP peer, and the LSR doesn't already have a mapping from the next hop.

   Note that since a non-merge LSR must setup a separate LSP for each upstream peer requesting a label, it must send a separate Label Request for each such peer. A consequence of this is that a non-merge LSR may have multiple Label Request messages for a given FEC outstanding at the same time.

The receiving LSR should respond to a Label Request message with a Label Mapping for the requested label or with a Notification message indicating why it cannot satisfy the request.

When the FEC for which a label is requested is a Prefix FEC Element or a Host Address FEC Element, the receiving LSR uses its routing table to determine its response. Unless its routing table includes an entry that exactly matches the requested Prefix or Host Address, the LSR must respond with a No Route Notification message.

The message ID of the Label Request message serves as an identifier for the Label Request transaction. When the receiving LSR responds with a Label Mapping message, the mapping message must include a Label Request/Returned Message ID TLV optional parameter which includes the message ID of the Label Request message. Note that since LSRs use Label Request message IDs as transaction identifiers an LSR should not reuse the message ID of a Label Request message until the corresponding transaction completes.

This version of the protocol defines the following Status Codes for the Notification message that signals a request cannot be satisfied:

No Route
  The FEC for which a label was requested includes a FEC Element for which the LSR does not have a route.

No Label Resources
  The LSR cannot provide a label because of resource limitations. When resources become available the LSR must notify the requesting LSR by sending a Notification message with the Label Resources Available Status Code.

  An LSR that receives a No Label Resources response to a Label Request message must not issue further Label Request messages until it receives a Notification message with the Label Resources Available Status code.

Loop Detected
  The LSR has detected a looping Label Request message.

See Appendix A "LDP Label Distribution Procedures" for more details.

3.5.9. Label Abort Request Message

The Label Abort Request message may be used to abort an outstanding Label Request message.

The encoding for the Label Abort 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Label Abort Req (0x0404)  |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message ID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      FEC TLV                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Label Request Message ID TLV             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Optional Parameters                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

FEC TLV
Identifies the FEC for which the Label Request is being aborted.

Label Request Message ID TLV
Specifies the message ID of the Label Request message to be aborted.

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

3.5.9.1. Label Abort Request Message Procedures

An LSR Ru may send a Label Abort Request message to abort an outstanding Label Request message for FEC sent to LSR Rd in the following circumstances:

1. Ru's next hop for FEC has changed from LSR Rd to LSR X; or

2. Ru is a non-merge, non-ingress LSR and has received a Label Abort Request for FEC from an upstream peer Y.

3. Ru is a merge, non-ingress LSR and has received a Label Abort Request for FEC from an upstream peer Y and Y is the only (last) upstream LSR requesting a label for FEC.

There may be other situations where an LSR may choose to abort an outstanding Label Request message in order to reclaim resource associated with the pending LSP. However, specification of general strategies for using the abort mechanism is beyond the scope of LDP.

When an LSR receives a Label Abort Request message, if it has not previously responded to the Label Request being aborted with a Label Mapping message or some other Notification message, it must acknowledge the abort by responding with a Label Request Aborted Notification message. The Notification must include a Label Request Message ID TLV that carries the message ID of the aborted Label Request message.

If an LSR receives a Label Abort Request Message after it has responded to the Label Request in question with a Label Mapping message or a Notification message, it ignores the abort request.

If an LSR receives a Label Mapping message in response to a Label Request message after it has sent a Label Abort Request message to abort the Label Request, the label in the Label Mapping message is valid. The LSR may choose to use the label or to release it with a Label Release message.

An LSR aborting a Label Request message may not reuse the Message ID for the Label Request message until it receives one of the following from its peer:

- A Label Request Aborted Notification message acknowledging the abort;

- A Label Mapping message in response to the Label Request message being aborted;

- A Notification message in response to the Label Request message being aborted (e.g., Loop Detected, No Label Resources, etc.).

To protect itself against tardy peers or faulty peer implementations an LSR may choose to time out receipt of the above. The time out period should be relatively long (several minutes). If the time out period elapses with no reply from the peer the LSR may reuse the Message Id of the Label Request message; if it does so, it should also discard any record of the outstanding Label Request and Label Abort messages.

Note that the response to a Label Abort Request message is never "ordered". That is, the response does not depend on the downstream state of the LSP setup being aborted. An LSR receiving a Label Abort Request message must process it immediately, regardless of the downstream state of the LSP, responding with a Label Request Aborted Notification or ignoring it, as appropriate.

3.5.10. 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0|   Label Withdraw (0x0402)   |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     FEC TLV                                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Label TLV (optional)                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Optional Parameters                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
32-bit value used to identify this message.

FEC TLV
Identifies the FEC for which the FEC-label mapping is being withdrawn.

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

Optional Parameter	Length	Value
Label TLV	variable	See below

The encoding for Label TLVs are found in Section "Label TLVs".

Label
If present, specifies the label being withdrawn (see procedures below).

3.5.10.1. Label Withdraw Message Procedures

An LSR transmits a Label Withdraw message under the following conditions:

1. The LSR no longer recognizes a previously known FEC for which it has advertised a label.

2. The LSR has decided unilaterally (e.g., via configuration) to no longer label switch a FEC (or FECs) with the label mapping being withdrawn.

The FEC TLV specifies the FEC for which labels are to be withdrawn. If no Label TLV follows the FEC, all labels associated with the FEC are to be withdrawn; otherwise only the label specified in the optional Label TLV is to be withdrawn.

The FEC TLV may contain the Wildcard FEC Element; if so, it may contain no other FEC Elements. In this case, if the Label Withdraw message contains an optional Label TLV, then the label is to be withdrawn from all FECs to which it is bound. If there is not an optional Label TLV in the Label Withdraw message, then the sending LSR is withdrawing all label mappings previously advertised to the receiving LSR.

An LSR that receives a Label Withdraw message must respond with a Label Release message.

See Appendix A "LDP Label Distribution Procedures" for more details.

3.5.11. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   Label Release (0x0403)   |      Message Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Message ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     FEC TLV                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Label TLV (optional)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Optional Parameters                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message ID
  32-bit value used to identify this message.

FEC TLV
Identifies the FEC for which the FEC-label mapping is being released.

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

Optional Parameter	Length	Value
Label TLV	variable	See below

The encodings for Label TLVs are found in Section "Label TLVs".

Label
If present, the label being released (see procedures below).

3.5.11.1. Label Release Message Procedures

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

An LSR must transmit a Label 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 receives a label mapping from an LSR which is not the next hop for the FEC, and the LSR is configured for conservative operation.

3. The LSR receives a Label Withdraw message.

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 TLV specifies the FEC for which labels are to be released. If no Label TLV follows the FEC, all labels associated with the FEC are to be released; otherwise only the label specified in the optional Label TLV is to be released.

The FEC TLV may contain the Wildcard FEC Element; if so, it may contain no other FEC Elements. In this case, if the Label Release message contains an optional Label TLV, then the label is to be released for all FECs to which it is bound. If there is not an optional Label TLV in the Label Release message, then the sending LSR is releasing all label mappings previously learned from the receiving LSR.

See Appendix A "LDP Label Distribution Procedures" for more details.

3.6. Messages and TLVs for Extensibility

Support for LDP extensibility includes the rules for the U and F bits that specify how an LSR should handle unknown TLVs and messages.

This section specifies TLVs and messages for vendor-private and experimental use.

3.6.1. LDP Vendor-private Extensions

Vendor-private TLVs and messages are used to convey vendor-private information between LSRs.

3.6.1.1. LDP Vendor-private TLVs

The Type range 0x3E00 through 0x3EFF is reserved for vendor-private TLVs.

The encoding for a vendor-private 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|    Type (0x3E00-0x3EFF)   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Vendor ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                           Data....                            |
~                                                               ~
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

U bit
Unknown TLV bit. Upon receipt of an unknown TLV, if U is clear (=0), a notification must be returned to the message originator and the entire message must be ignored; if U is set (=1), the unknown TLV is silently ignored and the rest of the message is processed as if the unknown TLV did not exist.

The determination as to whether a vendor-private message is understood is based on the Type and the mandatory Vendor ID field.

F bit
Forward unknown TLV bit. This bit only applies when the U bit is set and the LDP message containing the unknown TLV is is to be forwarded. If F is clear (=0), the unknown TLV is not forwarded with the containing message; if F is set (=1), the unknown TLV is forwarded with the containing message.

Type
Type value in the range 0x3E00 through 0x3EFF. Together, the Type and Vendor Id field specify how the Data field is to be interpreted.

Length
Specifies the cumulative length in octets of the Vendor ID and Data fields.

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.

3.6.1.2. LDP Vendor-private Messages

The Message Type range 0x3E00 through 0x3EFF is reserved for vendor-private Messages.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |U|    Msg Type (0x3E00-0x3EFF) |      Message Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Message ID                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Vendor ID                                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   +                                                               +
   |                     Remaining Mandatory Parameters            |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                     Optional Parameters                       |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

U bit
  Unknown message bit. Upon receipt of an unknown message, if U is clear (=0), a notification is returned to the message originator; if U is set (=1), the unknown message is silently ignored.

  The determination as to whether a vendor-private message is understood is based on the Msg Type and the Vendor ID parameter.

Msg Type
  Message type value in the range 0x3E00 through 0x3EFF. Together, the Msg Type and the Vendor ID specify how the message is to be interpreted.

Message Length
  Specifies the cumulative length in octets of the Message ID, Vendor ID, Remaining Mandatory Parameters and Optional Parameters.

Message ID
  32-bit 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".

Vendor ID
  802 Vendor ID as assigned by the IEEE.

Remaining Mandatory Parameters
  Variable length set of remaining required message parameters.

Optional Parameters
  Variable length set of optional message parameters.

3.6.2. LDP Experimental Extensions

LDP support for experimentation is similar to support for vendor-private extensions with the following differences:

- The Type range 0x3F00 through 0x3FFF is reserved for experimental TLVs.

- The Message Type range 0x3F00 through 0x3FFF is reserved for experimental messages.

- The encodings for experimental TLVs and messages are similar to the vendor-private encodings with the following difference.

  Experimental TLVs and messages use an Experiment ID field in place of a Vendor ID field. The Experiment ID field is used with the Type or Message Type field to specify the interpretation of the experimental TLV or Message.

  Administration of Experiment IDs is the responsibility of the experimenters.

3.7. Message Summary

The following are the LDP messages defined in this version of the protocol.

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	0x0400	"Label Mapping Message"
Label Request	0x0401	"Label Request Message"
Label Withdraw	0x0402	"Label Withdraw Message"
Label Release	0x0403	"Label Release Message"
Label Abort Request	0x0404	"Label Abort Request Message"
Vendor-Private	0x3E00-0x3EFF	"LDP Vendor-private Extensions"
Experimental	0x3F00-0x3FFF	"LDP Experimental Extensions"

3.8. 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"
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"
Returned PDU	0x0302	"Notification Message"
Returned Message	0x0303	"Notification Message"
Common Hello Parameters	0x0400	"Hello Message"
IPv4 Transport Address	0x0401	"Hello Message"
Configuration Sequence Number	0x0402	"Hello Message"
IPv6 Transport Address	0x0403	"Hello Message"
Common Session Parameters	0x0500	"Initialization Message"
ATM Session Parameters	0x0501	"Initialization Message"
Frame Relay Session Parameters	0x0502	"Initialization Message"

Label Request Message ID	0x0600	"Label Mapping Message"
Vendor-Private	0x3E00-0x3EFF	"LDP Vendor-private Extensions"
Experimental	0x3F00-0x3FFF	"LDP Experimental Extensions"

3.9. Status Code Summary

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

The "E" column is the required setting of the Status Code E-bit; the "Status Data" column is the value of the 30-bit Status Data field in the Status Code TLV.

Note that the setting of the Status Code F-bit is at the discretion of the LSR originating the Status TLV.

Status Code	E	Status Data	Section Title
Success	0	0x00000000	"Status TLV"
Bad LDP Identifier	1	0x00000001	"Events Signaled by ..."
Bad Protocol Version	1	0x00000002	"Events Signaled by ..."
Bad PDU Length	1	0x00000003	"Events Signaled by ..."
Unknown Message Type	0	0x00000004	"Events Signaled by ..."
Bad Message Length	1	0x00000005	"Events Signaled by ..."
Unknown TLV	0	0x00000006	"Events Signaled by ..."
Bad TLV length	1	0x00000007	"Events Signaled by ..."
Malformed TLV Value	1	0x00000008	"Events Signaled by ..."
Hold Timer Expired	1	0x00000009	"Events Signaled by ..."
Shutdown	1	0x0000000A	"Events Signaled by ..."
Loop Detected	0	0x0000000B	"Loop Detection"
Unknown FEC	0	0x0000000C	"FEC Procedures"
No Route	0	0x0000000D	"Label Request Mess ..."
No Label Resources	0	0x0000000E	"Label Request Mess ..."
Label Resources / Available	0	0x0000000F	"Label Request Mess ..."
Session Rejected/ No Hello	1	0x00000010	"Session Initialization"
Session Rejected/ Parameters Advertisement Mode	1	0x00000011	"Session Initialization"
Session Rejected/ Parameters Max PDU Length	1	0x00000012	"Session Initialization"
Session Rejected/ Parameters Label Range	1	0x00000013	"Session Initialization"

KeepAlive Timer Expired	1	0x00000014	"Events Signaled by ..."
Label Request Aborted	0	0x00000015	"Label Request Abort ..."
Missing Message Parameters	0	0x00000016	"Events Signaled by ..."
Unsupported Address Family	0	0x00000017	"FEC Procedures" "Address Message Proc ..."
Session Rejected/ Bad KeepAlive Time	1	0x00000018	"Session Initialization"
Internal Error	1	0x00000019	"Events Signaled by ..."

3.10. Well-known Numbers

3.10.1. UDP and TCP Ports

The UDP port for LDP Hello messages is 646.

The TCP port for establishing LDP session connections is 646.

3.10.2. Implicit NULL Label

The Implicit NULL label (see [ARCH]) is represented as a Generic Label TLV with a Label field value as specified by [ENCAP].

4. IANA Considerations

LDP defines the following name spaces which require management:

- Message types.
- TLV types.
- FEC types.
- Status codes.
- Experiment Ids.

The following sections provide guidelines for managing these name spaces.

4.1. Message Type Name Space

LDP divides the name space for message types into three ranges. The following are the guidelines for managing these ranges:

- Message Types 0x0000 - 0x3DFF. Message types in this range are part of the LDP base protocol. Following the policies outlined in [IANA], Message types in this range are allocated through an IETF Consensus action.

- Message Types 0x3E00 - 0x3EFF. Message types in this range are reserved for Vendor Private extensions and are the responsibility of the individual vendors (see Section "LDP Vendor-private Messages").

- Message Types 0x3F00 - 0x3FFF. Message types in this range are reserved for Experimental extensions and are the responsibility of the individual experimenters (see Sections "LDP Experimental Extensions" and "Experiment ID Name Space").

4.2. TLV Type Name Space

LDP divides the name space for TLV types into three ranges. The following are the guidelines for managing these ranges:

- TLV Types 0x0000 - 0x3DFF. TLV types in this range are part of the LDP base protocol. Following the policies outlined in [IANA], TLV types in this range are allocated through an IETF Consensus action.

- TLV Types 0x3E00 - 0x3EFF. TLV types in this range are reserved for Vendor Private extensions and are the responsibility of the individual vendors (see Section "LDP Vendor-private TLVs").

- TLV Types 0x3F00 - 0x3FFF. TLV types in this range are reserved for Experimental extensions and are the responsibility of the individual experimenters (see Sections "LDP Experimental Extensions" and "Experiment ID Name Space").

4.3. FEC Type Name Space

The range for FEC types is 0 - 255.

Following the policies outlined in [IANA], FEC types in the range 0 - 127 are allocated through an IETF Consensus action, types in the range 128 - 191 are allocated as First Come First Served, and types in the range 192 - 255 are reserved for Private Use.

4.4. Status Code Space

The range for Status Codes is 0x00000000 - 0x3FFFFFFF.

Following the policies outlined in [IANA], Status Codes in the range 0x00000000 - 0x1FFFFFFF are allocated through an IETF Consensus action, codes in the range 0x20000000 - 0x3EFFFFFF are allocated as First Come First Served, and codes in the range 0x3F000000 - 0x3FFFFFFF are reserved for Private Use.

4.5. Experiment ID Name Space

The range for Experiment Ids is 0x00000000 - 0xffffffff.

Following the policies outlined in [IANA], Experiment Ids in the range 0x00000000 - 0xefffffff are allocated as First Come First Served and Experiment Ids in the range 0xf0000000 are reserved for Private Use.

5. Security Considerations

This section identifies threats to which LDP may be vulnerable and discusses means by which those threats might be mitigated.

5.1. Spoofing

There are two types of LDP communication that could be the target of a spoofing attack.

1. Discovery exchanges carried by UDP.

   LSRs directly connected at the link level exchange Basic Hello messages over the link. The threat of spoofed Basic Hellos can be reduced by:

- o Accepting Basic Hellos only on interfaces to which LSRs that can be trusted are directly connected.

- o Ignoring Basic Hellos not addressed to the All Routers on this Subnet multicast group.

LSRs not directly connected at the link level may use Extended Hello messages to indicate willingness to establish an LDP session. An LSR can reduce the threat of spoofed Extended Hellos by filtering them and accepting only those originating at sources permitted by an access list.

2. Session communication carried by TCP.

LDP specifies use of the TCP MD5 Signature Option to provide for the authenticity and integrity of session messages.

[rfc2385] asserts that MD5 authentication is now considered by some to be too weak for this application. It also points out that a similar TCP option with a stronger hashing algorithm (it cites SHA-1 as an example) could be deployed. To our knowledge no such TCP option has been defined and deployed. However, we note that LDP can use whatever TCP message digest techniques are available, and when one stronger than MD5 is specified and implemented, upgrading LDP to use it would be relatively straightforward.

5.2. Privacy

LDP provides no mechanism for protecting the privacy of label distribution.

The security requirements of label distribution protocols are essentially identical to those of the protocols which distribute routing information. By providing a mechanism to ensure the authenticity and integrity of its messages LDP provides a level of security which is at least as good as, though no better than, that which can be provided by the routing protocols themselves. The more general issue of whether privacy should be required for routing protocols is beyond the scope of this document.

One might argue that label distribution requires privacy to address the threat of label spoofing. However, that privacy would not protect against label spoofing attacks since data packets carry labels in the clear. Furthermoe, label spoofing attacks can be made without knowledge of the FEC bound to a label.

To avoid label spoofing attacks, it is necessary to ensure that labeled data packets are labeled by trusted LSRs and that the labels placed on the packets are properly learned by the labeling LSRs.

5.3. Denial of Service

LDP provides two potential targets for denial of service (DoS) attacks:

1. Well known UDP Port for LDP Discovery

   An LSR adminstrator can address the threat of DoS attacks via Basic Hellos by ensuring that the LSR is directly connected only to peers which can be trusted to not initiate such an attack. Interfaces to peers interior to the administrator's domain should not represent a threat since interior peers are under the administrator's control. Interfaces to peers exterior to the domain represent a potential threat since exterior peers are not. An adminstrator can reduce that threat by connecting the LSR only to exterior peers that can be trusted to not initiate a Basic Hello attack.

   DoS attacks via Extended Hellos are potentially a more serious threat. This threat can be addressed by filtering Extended Hellos using access lists that define addresses with which extended discovery is permitted. However, performing the filering requires LSR resource.

   In an environment where a trusted MPLS cloud can be identified, LSRs at the edge of the cloud can be used to protect interior LSRs against DoS attacks via Extended Hellos by filtering out Extended Hellos originating outside of the trusted MPLS cloud, accepting only those originating at addresses permitted by access lists. This filtering protects LSRs in the interior of the cloud but consumes resoures at the edges.

2. Well known TCP port for LDP Session Establishment

   Like other control plane protocols that use TCP, LDP may be the target of DoS attacks, such a SYN attacks. LDP is no more or less vulnerable to such attacks than other control plane protocols that use TCP.

   The threat of such attacks can be mitigated somewhat by the following:

- o An LSR should avoid promiscuous TCP listens for LDP session establishment. It should use only listens that are specific to discovered peers. This enables it to drop attack packets early in their processing since they are less likely to match existing or in-progress connections.

- o The use of the MD5 option helps somewhat since it prevents a SYN from being accepted unless the MD5 segment checksum is valid. However, the receiver must compute the checksum before it can decide to discard an otherwise acceptable SYN segment.

- o The use of access list mechanisms applied at the boundary of the MPLS cloud in a manner similar to that suggested above for Extended Hellos can protect the interior against attacks originating from outside the cloud.

6. Areas for Future Study

The following topics not addressed in this version of LDP are possible areas for future study:

- Section 2.16 of the MPLS architecture [ARCH] requires that the initial label distribution protocol negotiation between peer LSRs enable each LSR to determine whether its peer is capable of popping the label stack. This version of LDP assumes that LSRs support label popping for all link types except ATM and Frame Relay. A future version may specify means to make this determination part of the session initiation negotiation.

- LDP support for CoS is not specified in this version. CoS support may be addressed in a future version.

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

- LDP support for multipath label switching is not specified in this version. Multipath support may be addressed in a future version.

7. Intellectual Property Considerations

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

8. 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 Gray, Yoshihiro Ohba, Eric Rosen, Bernard Suter, Yakov Rekhter, and Arun Viswanathan.

9. References

[ARCH] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", Work in Progress, July 1998.

[ATM] B. Davie, J. Lawrence, K. McCloghrie, Y. Rekhter, E. Rosen, G. Swallow, P. Doolan, "Use of Label Switching With ATM", Work in Progress, September, 1998.

[ATM-VP] N. Feldman, B. Jamoussi, S. Komandur, A, Viswanathan, T Worster, "MPLS using ATM VP Switching", Work in Progress, February, 1999.

[CRLDP] L. Andersson, A. Fredette, B. Jamoussi, R. Callon, P. Doolan, N. Feldman, E. Gray, J. Halpern, J. Heinanen T. E. Kilty, A. G. Malis, M. Girish, K. Sundell, P. Vaananen, T. Worster, L. Wu, R. Dantu, "Constraint-Based LSP Setup using LDP", Work in Progress, January, 1999.

[DIFFSERV] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", Work in Progress, October, 1998.

[ENCAP] E. Rosen, Y. Rekhter, D. Tappan, D. Farinacci, G. Fedorkow, T. Li, A. Conta, "MPLS Label Stack Encoding", Work in Progress, July, 1998.

[FR] A. Conta, P. Doolan, A. Malis, "Use of Label Switching on Frame Relay Networks", Work in Progress, October, 1998.

[FRAMEWORK] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, A. Viswanathan, "A Framework for Multiprotocol Label

Switching", Work in Progress, November 1997.

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

[LDPAPPLIC] B. Thomas, E. Gray, "LDP Applicability", Work in Progress, June 2000.

[LSPTUN] D. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, Vijay Srinivasan, "Extensions to RSVP for LSP Tunnels", Work in Progress, November 1998.

[rfc1321] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, April 1992.

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

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

[rfc1700] J. Reynolds, J.Postel, "ASSIGNED NUMBERS", October 1994.

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

[rfc2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[rfc2205] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[rfc2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

[TE] D. Awduche, J. Malcolm, J Agogbua, M. O'Dell, J. McManus, " Requirements for Traffic Engineering over MPLS", Work in Progress, October 1998.

10. Author Information

Loa Andersson
Nortel Networks Inc
St Eriksgatan 115, PO Box 6701
113 85 Stockholm
Sweden
Phone: +46 8 5088 36 34
Mobile: +46 70 522 78 34
email: loa.andersson@nortelnetworks.com

Andre Fredette
Nortel Networks Inc
600 Technology Park Drive
Billerica, MA 01821
Phone: 978-288-8524
email: fredette@nortelnetworks.com

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

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

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

Appendix A. LDP Label Distribution Procedures

This section specifies label distribution behavior in terms of LSR response to the following events:

- Receive Label Request Message;
- Receive Label Mapping Message;
- Receive Label Abort Request Message;
- Receive Label Release Message;
- Receive Label Withdraw Message;
- Recognize new FEC;
- Detect change in FEC next hop;
- Receive Notification Message / Label Request Aborted;
- Receive Notification Message / No Label Resources;
- Receive Notification Message / No Route;
- Receive Notification Message / Loop Detected;
- Receive Notification Message / Label Resources Available;
- Detect local label resources have become available;
- LSR decides to no longer label switch a FEC;
- Timeout of deferred label request.

The specification of LSR behavior in response to an event has three parts:

1. Summary. Prose that describes LSR response to the event in overview.

2. Context. A list of elements referred to by the Algorithm part of the specification. (See 3.)

3. Algorithm. An algorithm for LSR response to the event.

The Summary may omit details of the LSR response, such as bookkeeping action or behavior dependent on the LSR label advertisement mode, control mode, or label retention mode in use. The intent is that the Algorithm fully and unambiguously specify the LSR response.

The algorithms in this section use procedures defined in the MPLS architecture specification [ARCH] for hop-by-hop routed traffic. These procedures are:

- Label Distribution procedure, which is performed by a downstream LSR to determine when to distribute a label for a FEC to LDP peers. The architecture defines four Label Distribution procedures:

 . Downstream Unsolicited Independent Control, called PushUnconditional in [ARCH].

 . Downstream Unsolicited Ordered Control, called PushConditional in [ARCH].

 . Downstream On Demand Independent Control, called PulledUnconditional in [ARCH].

 . Downstream On Demand Ordered Control, called PulledConditional in [ARCH].

- Label Withdrawal procedure, which is performed by a downstream LSR to determine when to withdraw a FEC label mapping previously distributed to LDP peers. The architecture defines a single Label Withdrawal procedure. Whenever an LSR breaks the binding between a label and a FEC, it must withdraw the FEC label mapping from all LDP peers to which it has previously sent the mapping.

- Label Request procedure, which is performed by an upstream LSR to determine when to explicitly request that a downstream LSR bind a label to a FEC and send it the corresponding label mapping. The architecture defines three Label Request procedures:

 . Request Never. The LSR never requests a label.

 . Request When Needed. The LSR requests a label whenever it needs one.

 . Request On Request. This procedure is used by non-label merging LSRs. The LSR requests a label when it receives a request for one, in addition to whenever it needs one.

- Label Release procedure, which is performed by an upstream LSR to determine when to release a previously received label mapping for a FEC. The architecture defines two Label Release procedures:

 . Conservative label retention, called Release On Change in [ARCH].

 . Liberal label retention, called No Release On Change in [ARCH].

- Label Use procedure, which is performed by an LSR to determine when to start using a FEC label for forwarding/switching. The architecture defines three Label Use procedures:

. Use Immediate. The LSR immediately uses a label received from a FEC next hop for forwarding/switching.

. Use If Loop Free. The LSR uses a FEC label received from a FEC next hop for forwarding/switching only if it has determined that by doing so it will not cause a forwarding loop.

. Use If Loop Not Detected. This procedure is the same as Use Immediate unless the LSR has detected a loop in the FEC LSP. Use of the FEC label for forwarding/switching will continue until the next hop for the FEC changes or the loop is no longer detected.

This version of LDP does not include a loop prevention mechanism; therefore, the procedures below do not make use of the Use If Loop Free procedure.

- Label No Route procedure (called Label Not Available procedure in [ARCH]), which is performed by an upstream LSR to determine how to respond to a No Route notification from a downstream LSR in response to a request for a FEC label mapping. The architecture specification defines two Label No Route procedures:

  . Request Retry. The LSR should issue the label request at a later time.

  . No Request Retry. The LSR should assume the downstream LSR will provide a label mapping when the downstream LSR has a next hop and it should not reissue the request.

A.1. Handling Label Distribution Events

This section defines LDP label distribution procedures by specifying an algorithm for each label distribution event. The requirement on an LDP implementation is that its event handling must have the effect specified by the algorithms. That is, an implementation need not follow exactly the steps specified by the algorithms as long as the effect is identical.

The algorithms for handling label distribution events share common actions. The specifications below package these common actions into procedure units. Specifications for these common procedures are in their own section "Common Label Distribution Procedures", which follows this.

An implementation would use data structures to store information

about protocol activity. This appendix specifies the information to be stored in sufficient detail to describe the algorithms, and assumes the ability to retrieve the information as needed. It does not specify the details of the data structures.

A.1.1. Receive Label Request

Summary:

The response by an LSR to receipt of a FEC label request from an LDP peer may involve one or more of the following actions:

- Transmission of a notification message to the requesting LSR indicating why a label mapping for the FEC cannot be provided;

- Transmission of a FEC label mapping to the requesting LSR;

- Transmission of a FEC label request to the FEC next hop;

- Installation of labels for forwarding/switching use by the LSR.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.

- FEC. The FEC specified in the message.

- RAttributes. Attributes received with the message. E.g., Hop Count Path Vector.

- SAttributes. Attributes to be included in Label Request message, if any, propagated to FEC Next Hop.

- StoredHopCount. The hop count, if any, previously recorded for the FEC.

Algorithm:

```
LRq.1   Execute procedure Check_Received_Attributes (MsgSource,
        LabelRequest, RAttributes).
        If Loop Detected, goto LRq.13.

LRq.2   Is there a Next Hop for FEC?
        If not, goto LRq.5.
```

```
   LRq.3    Is MsgSource the Next Hop?
            Ifnot, goto LRq.6.

   LRq.4    Execute procedure Send_Notification (MsgSource, Loop
            Detected).
            Goto LRq.13

   LRq.5    Execute procedure Send_Notification (MsgSource, No Route).
            Goto LRq.13.

   LRq.6    Has LSR previously received a label request for FEC from
            MsgSource?
            If not, goto LRq.8.  (See Note 1.)

   LRq.7    Is the label request a duplicate request?
            If so, Goto LRq.13.  (See Note 2.)

   LRq.8    Record label request for FEC received from MsgSource and mark
            it pending.

   LRq.9    Perform LSR Label Distribution procedure:

              For Downstream Unsolicited Independent Control OR
              For Downstream On Demand Independent Control

                1.   Has LSR previously received and retained a label
                     mapping for FEC from Next Hop?.
                     Is so, set Propagating to IsPropagating.
                     If not, set Propagating to NotPropagating.

                2.   Execute procedure
                     Prepare_Label_Mapping_Attributes(MsgSource, FEC,
                     RAttributes, SAttributes, Propagating,
                     StoredHopCount).

                3.   Execute procedure Send_Label (MsgSource, FEC,
                     SAttributes).

                4.   Is LSR egress for FEC? OR
                     Has LSR previously received and retained a label
                     mapping for FEC from Next Hop?
                     If so, goto LRq.11.  If not, goto LRq.10.

              For Downstream Unsolicited Ordered Control OR
              For Downstream On Demand Ordered Control

                1.   Is LSR egress for FEC? OR
                     Has LSR previously received and retained a label
```

```
                   mapping for FEC from Next Hop?  (See Note 3.)
                   If not, goto LRq.10.

               2.  Execute procedure
                   Prepare_Label_Mapping_Attributes(MsgSource, FEC,
                   RAttributes, SAttributes, IsPropagating,
                   StoredHopCount)

               3.  Execute procedure Send_Label (MsgSource, FEC,
                   SAttributes).
                   Goto LRq.11.

   LRq.10  Perform LSR Label Request procedure:

             For Request Never

               1.  Goto LRq.13.

             For Request When Needed OR
             For Request On Request

               1.  Execute procedure Prepare_Label_Request_Attributes
                   (Next Hop, FEC, RAttributes, SAttributes);

               2.  Execute procedure Send_Label_Request (Next Hop, FEC,
                   SAttributes).
                   Goto LRq.13.

   LRq.11  Has LSR successfully sent a label for FEC to MsgSource?
           If not, goto LRq.13.  (See Note 4.)

   LRq.12  Perform LSR Label Use procedure.

             For Use Immediate OR
             For Use If Loop Not Detected

               1.  Install label sent to MsgSource and label from Next
                   Hop (if LSR is not egress) for forwarding/switching
                   use.

   LRq.13  DONE
```

Notes:

1. In the case where MsgSource is a non-label merging LSR it will send a label request for each upstream LDP peer that has requested a label for FEC from it. The LSR must be able to distinguish such requests from a non-label merging MsgSource

from duplicate label requests.

The LSR uses the message ID of received Label Request messages to detect duplicate requests. This means that an LSR (the upstream peer) may not reuse the message ID used for a Label Request until the Label Request transaction has completed.

2. When an LSR sends a label request to a peer it records that the request has been sent and marks it as outstanding. As long as the request is marked outstanding the LSR should not send another request for the same label to the peer. Such a second request would be a duplicate. The Send_Label_Request procedure described below obeys this rule.

   A duplicate label request is considered a protocol error and should be dropped by the receiving LSR (perhaps with a suitable notification returned to MsgSource).

3. If LSR is not merge-capable, this test will fail.

4. The Send_Label procedure may fail due to lack of label resources, in which case the LSR should not perform the Label Use procedure.

A.1.2. Receive Label Mapping

Summary:

The response by an LSR to receipt of a FEC label mapping from an LDP peer may involve one or more of the following actions:

- Transmission of a label release message for the FEC label to the LDP peer;

- Transmission of label mapping messages for the FEC to one or more LDP peers,

- Installation of the newly learned label for forwarding/switching use by the LSR.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.

- FEC. The FEC specified in the message.

- Label. The label specified in the message.

- PrevAdvLabel. The label for FEC, if any, previously advertised to an upstream peer.

- StoredHopCount. The hop count previously recorded for the FEC.

- RAttributes. Attributes received with the message. E.g., Hop Count, Path Vector.

- SAttributes to be included in Label Mapping message, if any, propagated to upstream peers.

Algorithm:

```
  LMp.1   Does the received label mapping match an outstanding label
          request for FEC previously sent to MsgSource.
          If not, goto LMp.3.

  LMp.2   Delete record of outstanding FEC label request.

  LMp.3   Execute procedure Check_Received_Attributes (MsgSource,
          LabelMapping, RAttributes).
          If No Loop Detected, goto LMp.9.

  LMp.4   Does the LSR have a previously received label mapping for FEC
          from MsgSource? (See Note 1.)
          If not, goto LMp.8. (See Note 2.)

  LMp.5   Does the label previously received from MsgSource match Label
          (i.e., the label received in the message)? (See Note 3.)
          If not, goto LMp.8. (See Note 4.)

  LMp.6   Delete matching label mapping for FEC previously received
          from MsgSource.

  LMp.7   Remove Label from forwarding/switching use. (See Note 5.)
          Goto LMp.33.

  LMp.8   Execute procedure Send_Message (MsgSource, Label Release,
          FEC, Label, Loop Detected Status code).  Goto LMp.33.

  LMp.9   Does LSR have a previously received label mapping for FEC
          from MsgSource for the LSP in question?  (See Note 6.)
```

```
           If not, goto LMp.11.

   LMp.10  Does the label previously received from MsgSource match Label
           (i.e., the label received in the message)?  (See Note 3.)
           If not, goto LMp.32.  (See Note 4.)

   LMp.11  Determine the Next Hop for FEC.

   LMp.12  Is MsgSource the Next Hop for FEC?
           If so, goto LMp.14.

   LMp.13  Perform LSR Label Release procedure:

             For Conservative Label retention:

               1.  Goto LMp.32.

             For Liberal Label retention:

               1.  Record label mapping for FEC with Label and
                   RAttributes has been received from MsgSource.
                   Goto LMp.33.

   LMp.14  Is LSR an ingress for FEC?
           If not, goto LMp.16.

   LMp.15  Install Label for forwarding/switching use.

   LMp.16  Record label mapping for FEC with Label and RAttributes has
           been received from MsgSource.

   LMp.17  Iterate through LMp.31 for each Peer.  (See Note 7).

   LMp.18  Has LSR previously sent a label mapping for FEC to Peer for
           the LSP in question?  (See Note 8.)
           If so, goto LMp.22.

   LMp.19  Is the Downstream Unsolicited Ordered Control Label
           Distribution procedure being used by LSR?  If not, goto
           LMp.28.

   LMp.20  Execute procedure Prepare_Label_Mapping_Attributes(Peer, FEC,
           RAttributes, SAttributes, IsPropagating, StoredHopCount).

   LMp.21  Execute procedure Send_Message (Peer, Label Mapping, FEC,
           PrevAdvLabel, SAttributes).
           Goto LMp.28
```

```
   LMp.22  Iterate through LMp.27 for each label mapping for FEC
           previously sent to Peer.

   LMp.23  Are RAttributes in the received label mapping consistent with
           those previously sent to Peer?
           If so, continue iteration from LMp.22 for next label mapping.
           (See Note 9.)

   LMp.24  Execute procedure Prepare_Label_Mapping_Attributes(Peer, FEC,
           RAttributes, SAttributes, IsPropagating, StoredHopCount).

   LMp.25  Execute procedure Send_Message (Peer, Label Mapping, FEC,
           PrevAdvLabel, SAttributes).  (See Note 10.)

   LMp.26  Update record of label mapping for FEC previously sent to
           Peer to include the new attributes sent.

   LMp.27  End iteration from LMp.22.

   LMp.28  Does LSR have any label requests for FEC from Peer marked as
           pending?
           If not, goto LMp.30.

   LMp.29  Perform LSR Label Distribution procedure:

             For Downstream Unsolicited Independent Control OR
             For Downstream Unsolicited Ordered Control

               1.  Execute procedure
                   Prepare_Label_Mapping_Attributes(Peer, FEC,
                   RAttributes, SAttributes, IsPropagating,
                   UnknownHopCount).

               2.  Execute procedure Send_Label (Peer, FEC,
                   SAttributes).
                   If the procedure fails, continue iteration for next
                   Peer at LMp.17.

               3.  If no pending requests exist for Peer goto LMp.30.
                   (See Note 11.)

             For Downstream On Demand Independent Control OR
             For Downstream On Demand Ordered Control

               1.  Iterate through Step 5 for each pending label request
                   for FEC from Peer marked as pending.
```

```
               2.  Execute procedure
                   Prepare_Label_Mapping_Attributes(Peer, FEC,
                   RAttributes, SAttributes, IsPropagating,
                   UnknownHopCount)

               3.  Execute procedure Send_Label (Peer, FEC,
                   SAttributes).
                   If the procedure fails, continue iteration for next
                   Peer at LMp.17.

               4.  Delete record of pending request.

               5.  End iteration from Step 1.

               6.  Goto LMp.30.

   LMp.30  Perform LSR Label Use procedure:

             For Use Immediate OR
             For Use If Loop Not Detected

               1.  Iterate through Step 3 for each label mapping for FEC
                   previously sent to Peer.

               2.  Install label received and label sent to Peer for
                   forwarding/switching use.

               3.  End iteration from Step 1.

               4.  Goto LMp.31.

   LMp.31  End iteration from LMp.17.
           Go to LMp.33.

   LMp.32  Execute procedure Send_Message (MsgSource, Label Release,
           FEC, Label).

   LMp.33  DONE.
```

Notes:

1. If the LSR is merging there should be at most 1 received mapping for the FEC for the LSP in question. In the non-merging case there could be multiple received mappings for the FEC for the LSP in question.

2. If LSR has detected a loop and it has not previously received a label mapping from MsgSource for the FEC, it simply releases the label.

3. Does the Label received in the message match any of the 1 or more label mappings identified in the previous step (LMp.4 or LMp.9)?

4. An unsolicited mapping with a different label from the same peer would be an attempt to establish multipath label switching, which is not supported in this version of LDP.

5. If Label is not in forwarding/switching use, LMp.7 has no effect.

6. If the received label mapping message matched an outstanding label request in LMp.1, then (by definition) LSR has not previously received a label mapping for FEC for the LSP in question. If the LSR is merging upstream labels for the LSP in question, there should be at most 1 received mapping. In the non-merging case, there could be multiple received label mappings for the same FEC, one for each resulting LSP.

7. The LMp.17 iteration includes MsgSource in order to handle the case where LSR is operating in Downstream Unsolicited ordered control mode. Ordered control prevents LSR from advertising a label for FEC until it has received a label mapping from its next hop (MsgSource) for FEC.

8. If LSR is merging the LSP it may have previously sent label mappings for the FEC LSP to one or more peers. If LSR is not merging, it may have sent a label mapping for the LSP in question to at most one LSR.

9. The loop detection Path Vector attribute is considered in this check. If the received RAttributes include a Path Vector and no Path Vector had been previously sent to the Peer, or if the received Path Vector is inconsistent with the Path Vector previously sent to the Peer, then the attributes are considered to be inconsistent. Note that an LSR is not required to store a received Path Vector after it propagates the Path Vector in a mapping message. If an LSR does not store the Path Vector, it has no way to check the consistency of a newly received Path Vector. This means that whenever such an LSR receives a mapping message carrying a Path Vector it must always propagate the Path Vector.

10. LMp.22 through LMp.27 deal with a situation that can arise when the LSR is using independent control and it receives a mapping from the downstream peer after it has sent a mapping to an upstream peer. In this situation the LSR needs to propagate any changed attributes, such as Hop Count, upstream. If Loop Detection is configured on, the propagated attributes must include the Path Vector

11. An LSR operating in Downstream Unsolicited mode must process any Label Request messages it receives. If there are pending label requests, fall through into the Downstream on Demand procedures in order to satisfy the pending requests.

A.1.3. Receive Label Abort Request

Summary:

When an LSR receives a label abort request message from a peer, it checks whether it has already responded to the label request in question. If it has, it silently ignores the message. If it has not, it sends the peer a Label Request Aborted Notification. In addition, if it has a label request outstanding for the LSP in question to a downstream peer, it sends a Label Abort Request to the downstream peer to abort the LSP.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.

- FEC. The FEC specified in the message.

- RequestMessageID. The message ID of the label request message to be aborted.

- Next Hop. The next hop for the FEC.

Algorithm:

```
  LAbR.1  Does the message match a previously received label request
          message from MsgSource? (See Note 1.)
          If not, goto LAbR.12.

  LAbR.2  Has LSR responded to the previously received label request?
          If so, goto LAbR.12.
```

LAbR.3 Execute procedure Send_Message(MsgSource, Notification, Label Request Aborted, TLV), where TLV is the Label Request Message ID TLV received in the label abort request message.

LAbR.4 Does LSR have a label request message outstanding for FEC?
If so, goto LAbR.7

LAbR.5 Does LSR have a label mapping for FEC?
If not, goto LAbR.11

LAbR.6 Generate Event: Received Label Release Message for FEC from MsgSource. (See Note 2.)
Goto LAbR.11.

LAbR.7 Is LSR merging the LSP for FEC?
If not, goto LAbR.9.

LAbR.8 Are there upstream peers other than MsgSource that have requested a label for FEC?
If so, goto LAbR.11.

LAbR.9 Execute procedure Send_Message (Next Hop, Label Abort Request, FEC, TLV), where TLV is a Label Request Message ID TLV containing the Message ID used by the LSR in the outstanding Label Request message.

LAbR.10 Record that a label abort request for FEC is pending.

LAbR.11 Delete record of label request for FEC from MsgSource.

LAbR.12 DONE

Notes:

1. LSR uses FEC and the Label Request Message ID TLV carried by the label abort request to locate its record (if any) for the previously received label request from MsgSource.

2. If LSR has received a label mapping from NextHop, it should behave as if it had advertised a label mapping to MsgSource and MsgSource has released it.

A.1.4. Receive Label Release

Summary:

When an LSR receives a label release message for a FEC from a peer, it checks whether other peers hold the released label. If none do, the LSR removes the label from forwarding/switching use, if it has not already done so, and if the LSR holds a label mapping from the FEC next hop, it releases the label mapping.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.

- Label. The label specified in the message.

- FEC. The FEC specified in the message.

Algorithm:

```
LRl.1   Remove MsgSource from record of peers that hold Label for
        FEC.  (See Note 1.)

LRl.2   Does message match an outstanding label withdraw for FEC
        previously sent to MsgSource?
        If not, goto LRl.4

LRl.3   Delete record of outstanding label withdraw for FEC
        previously sent to MsgSource.

LRl.4   Is LSR merging labels for this FEC?
        If not, goto LRl.6.  (See Note 2.)

LRl.5   Has LSR previously advertised a label for this FEC to other
        peers?
        If so, goto LRl.10.

LRl.6   Is LSR egress for the FEC?
        If so, goto LRl.10

LRl.7   Is there a Next Hop for FEC? AND
        Does LSR have a previously received label mapping for FEC
        from Next Hop?
        If not, goto LRl.10.

LRl.8   Is LSR configured to propagate releases?
```

```
           If not, goto LRl.10.  (See Note 3.)

   LRl.9   Execute procedure Send_Message (Next Hop, Label Release, FEC,
           Label from Next Hop).

   LRl.10  Remove Label from forwarding/switching use for traffic from
           MsgSource.

   LRl.11  Do any peers still hold Label for FEC?
           If so, goto LRl.13.

   LRl.12  Free the Label.

   LRl.13  DONE.
```

Notes:

1. If LSR is using Downstream Unsolicited label distribution, it should not re-advertise a label mapping for FEC to MsgSource until MsgSource requests it.

2. LRl.4 through LRl.8 deal with determining whether where the LSR should propagate the label release to a downstream peer (LRl.9).

3. If LRl.8 is reached, no upstream LSR holds a label for the FEC, and the LSR holds a label for the FEC from the FEC Next Hop. The LSR could propagate the Label Release to the Next Hop. By propagating the Label Release the LSR releases a potentially scarce label resource. In doing so, it also increases the latency for re-establishing the LSP should MsgSource or some other upstream LSR send it a new Label Request for FEC.

   Whether or not to propagate the release is not a protocol issue. Label distribution will operate properly whether or not the release is propagated. The decision to propagate or not should take into consideration factors such as: whether labels are a scarce resource in the operating environment; the importance of keeping LSP setup latency low by keeping the amount of signaling required small; whether LSP setup is ingress-controlled or egress-controlled in the operating environment.

A.1.5. Receive Label Withdraw

Summary:

When an LSR receives a label withdraw message for a FEC from an LDP peer, it responds with a label release message and it removes the label from any forwarding/switching use. If ordered control is in use, the LSR sends a label withdraw message to each LDP peer to which it had previously sent a label mapping for the FEC. If the LSR is using Downstream on Demand label advertisement with independent control, it then acts as if it had just recognized the FEC.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.

- Label. The label specified in the message.

- FEC. The FEC specified in the message.

Algorithm:

```
LWd.1   Remove Label from forwarding/switching use.  (See Note 1.)

LWd.2   Execute procedure Send_Message (MsgSource, Label Release,
        FEC, Label)

LWd.3   Has LSR previously received and retained a matching label
        mapping for FEC from MsgSource?
        If not, goto LWd.13.

LWd.4   Delete matching label mapping for FEC previously received
        from MsgSource.

LWd.5   Is LSR using ordered control?
        If so, goto LWd.8.

LWd.6   Is MsgSource using Downstream On Demand label advertisement?
        If not, goto LWd.13.

LWd.7   Generate Event: Recognize New FEC for FEC.
        Goto LWd.13.  (See Note 2.)

LWd.8   Iterate through LWd.12 for each Peer, other than MsgSource.
```

```
   LWd.9    Has LSR previously sent a label mapping for FEC to Peer?
            If not, continue iteration for next Peer at LWd.8.

   LWd.10   Does the label previously sent to Peer "map" to the withdrawn
            Label?
            If not, continue iteration for next Peer at LWd.8.  (See Note
            3.)

   LWd.11   Execute procedure Send_Label_Withdraw (Peer, FEC, Label
            previously sent to Peer).

   LWd.12   End iteration from LWd.8.

   LWd.13   DONE
```

Notes:

1. If Label is not in forwarding/switching use, LWd.1 has no effect.

2. LWd.7 handles the case where the LSR is using Downstream On Demand label distribution with independent control. In this situation the LSR should send a label request to the FEC next hop as if it had just recognized the FEC.

3. LWd.10 handles both label merging (one or more incoming labels map to the same outgoing label) and no label merging (one label maps to the outgoing label) cases.

A.1.6. Recognize New FEC

Summary:

The response by an LSR to learning a new FEC via the routing table may involve one or more of the following actions:

- Transmission of label mappings for the FEC to one or more LDP peers;

- Transmission of a label request for the FEC to the FEC next hop;

- Any of the actions that can occur when the LSR receives a label mapping for the FEC from the FEC next hop.

Context:

- LSR. The LSR handling the event.

- FEC. The newly recognized FEC.

- Next Hop. The next hop for the FEC.

- InitAttributes. Attributes to be associated with the new FEC. (See Note 1.)

- SAttributes. Attributes to be included in Label Mapping or Label Request messages, if any, sent to peers.

- StoredHopCount. Hop count associated with FEC label mapping, if any, previously received from Next Hop.

Algorithm:

```
  FEC.1   Perform LSR Label Distribution procedure:

            For Downstream Unsolicited Independent Control

              1.  Iterate through 5 for each Peer.

              2.  Has LSR previously received and retained a label
                  mapping for FEC from Next Hop?
                  If so, set Propagating to IsPropagating.
                  If not, set Propagating to NotPropagating.

              3.  Execute procedure Prepare_Label_Mapping_Attributes
                  (Peer, FEC, InitAttributes, SAttributes, Propagating,
                  Unknown hop count(0)).

              4.  Execute procedure Send_Label (Peer, FEC, SAttributes)

              5.  End iteration from 1.
                  Goto FEC.2.

            For Downstream Unsolicited Ordered Control

              1.  Iterate through 5 for each Peer.

              2.  Is LSR egress for the FEC? OR
                  Has LSR previously received and retained a label
                  mapping for FEC from Next Hop?
                  If not, continue iteration for next Peer.
```

```
              3.   Execute procedure Prepare_Label_Mapping_Attributes
                   (Peer, FEC, InitAttributes, SAttributes, Propagating,
                   StoredHopCount).

              4.   Execute procedure Send_Label (Peer, FEC, SAttributes)

              5.   End iteration from 1.
                   Goto FEC.2.

            For Downstream On Demand Independent Control OR
            For Downstream On Demand Ordered Control

              1.   Goto FEC.2.  (See Note 2.)

   FEC.2   Has LSR previously received and retained a label mapping for
           FEC from Next Hop?
           If so, goto FEC.5

   FEC.3   Is Next Hop an LDP peer?
           If not, Goto FEC.6

   FEC.4   Perform LSR Label Request procedure:

            For Request Never

              1.   Goto FEC.6

            For Request When Needed OR
            For Request On Request

              1.   Execute procedure Prepare_Label_Request_Attributes
                   (Next Hop, FEC, InitAttributes, SAttributes);

              2.   Execute procedure Send_Label_Request (Next Hop, FEC,
                   SAttributes).
                   Goto FEC.6.

   FEC.5   Generate Event: Received Label Mapping from Next Hop.  (See
           Note 3.)

   FEC.6   DONE.
```

Notes:

1. An example of an attribute that might be part of InitAttributes is one which specifies desired LSP characteristics, such as class of service (CoS). (Note that while the current version of LDP does not specify a CoS attribute, LDP extensions may.)

The means by which FEC InitAttributes, if any, are specified is beyond the scope of LDP. Note that the InitAttributes will not include a known Hop Count or a Path Vector.

2. An LSR using Downstream On Demand label distribution would send a label only if it had a previously received label request marked as pending. The LSR would have no such pending requests because it responds to any label request for an unknown FEC by sending the requesting LSR a No Route notification and discarding the label request; see LRq.3

3. If the LSR has a label for the FEC from the Next Hop, it should behave as if it had just received the label from the Next Hop. This occurs in the case of Liberal label retention mode.

A.1.7. Detect Change in FEC Next Hop

Summary:

The response by an LSR to a change in the next hop for a FEC may involve one or more of the following actions:

- Removal of the label from the FEC's old next hop from forwarding/switching use;

- Transmission of label mapping messages for the FEC to one or more LDP peers;

- Transmission of a label request to the FEC's new next hop;

- Any of the actions that can occur when the LSR receives a label mapping from the FEC's new next hop.

Context:

- LSR. The LSR handling the event.

- FEC. The FEC whose next hop changed.

- New Next Hop. The current next hop for the FEC.

- Old Next Hop. The previous next hop for the FEC.

- OldLabel. Label, if any, previously received from Old Next Hop.

- CurAttributes. The attributes, if any, currently associated with the FEC.

- SAttributes. Attributes to be included in Label Label Request message, if any, sent to New Next Hop.

Algorithm:

```
NH.1   Has LSR previously received and retained a label mapping for
       FEC from Old Next Hop?
       If not, goto NH.6.

NH.2   Remove label from forwarding/switching use. (See Note 1.)

NH.3   Is LSR using Liberal label retention?
       If so, goto NH.6.

NH.4   Execute procedure Send_Message (Old Next Hop, Label Release,
       OldLabel).

NH.5   Delete label mapping for FEC previously received from Old
       Next Hop.

NH.6   Does LSR have a label request pending with Old Next Hop?
       If not, goto NH.10.

NH.7   Is LSR using Conservative label retention?
       If not, goto NH.10.

NH.8   Execute procedure Send_Message (Old Next Hop, Label Abort
       Request, FEC, TLV), where TLV is a Label Request Message ID
       TLV that carries the message ID of the pending label request.

NH.9   Record a label abort request is pending for FEC with Old Next
       Hop.

NH.10  Is there a New Next Hop for the FEC?
       If not, goto NH.16.

NH.11  Has LSR previously received and retained a label mapping for
       FEC from New Next Hop?
       If not, goto NH.13.

NH.12  Generate Event: Received Label Mapping from New Next Hop.
       Goto NH.20. (See Note 2.)

NH.13  Is LSR using Downstream on Demand advertisement? OR
       Is Next Hop using Downstream on Demand advertisement? OR
```

```
              Is LSR using Conservative label retention? (See Note 3.)
              If so, goto NH.14.
              If not, goto NH.20.

       NH.14  Execute procedure Prepare_Label_Request_Attributes (Next Hop,
              FEC, CurAttributes, SAttributes)

       NH.15  Execute procedure Send_Label_Request (New Next Hop, FEC,
              SAttributes).  (See Note 4.)
              Goto NH.20.

       NH.16  Iterate through NH.19 for each Peer.

       NH.17  Has LSR previously sent a label mapping for FEC to Peer?
              If not, continue iteration for next Peer at NH.16.

       NH.18  Execute procedure Send_Label_Withdraw (Peer, FEC, Label
              previously sent to Peer).

       NH.19  End iteration from NH.16.

       NH.20  DONE.
```

Notes:

1. If Label is not in forwarding/switching use, NH.2 has no effect.

2. If the LSR has a label for the FEC from the New Next Hop, it should behave as if it had just received the label from the New Next Hop.

3. The purpose of the check on label retention mode is to avoid a race with steps LMp.12-LMp.13 of the procedure for handling a Label Mapping message where the LSR operating in Conservative Label retention mode may have released a label mapping received from the New Next Hop before it detected the FEC next hop had changed.

4. Regardless of the Label Request procedure in use by the LSR, it must send a label request if the conditions in NH.8 hold. Therefore it executes the Send_Label_Request procedure directly rather than perform LSR Label Request procedure.

A.1.8. Receive Notification / Label Request Aborted

Summary:

When an LSR receives a Label Request Aborted notification from an LDP peer it records that the corresponding label request transaction, if any, has completed.

Context:

- LSR. The LSR handling the event.

- FEC. The FEC for which a label was requested.

- RequestMessageID. The message ID of the label request message to be aborted.

- MsgSource. The LDP peer that sent the Notification message.

Algorithm:

```
LRqA.1  Does the notification correspond to an outstanding label
        request abort for FEC? (See Note 1).
        If not, goto LRqA.3.

LRqA.2  Record that the label request for FEC has been aborted.

LRqA.3  DONE
```

Notes:

1. The LSR uses the FEC and RequestMessageID to locate its record, if any, of the outstanding label request abort.

A.1.9. Receive Notification / No Label Resources

Summary:

When an LSR receives a No Label Resources notification from an LDP peer, it stops sending label request messages to the peer until it receives a Label Resources Available Notification from the peer.

Context:

- LSR. The LSR handling the event.

- FEC. The FEC for which a label was requested.

- MsgSource. The LDP peer that sent the Notification message.

Algorithm:

```
NoRes.1 Delete record of outstanding label request for FEC sent to
        MsgSource.

NoRes.2 Record label mapping for FEC from MsgSource is needed but
        that no label resources are available.

NoRes.3 Set status record indicating it is not OK to send label
        requests to MsgSource.

NoRes.4 DONE.
```

A.1.10. Receive Notification / No Route

Summary:

When an LSR receives a No Route notification from an LDP peer in response to a Label Request message, the Label No Route procedure in use dictates its response. The LSR either will take no further action, or it will defer the label request by starting a timer and send another Label Request message to the peer when the timer later expires.

Context:

- LSR. The LSR handling the event.

- FEC. The FEC for which a label was requested.

- Attributes. The attributes associated with the label request.

- MsgSource. The LDP peer that sent the Notification message.

Algorithm:

```
  NoNH.1  Delete record of outstanding label request for FEC sent to
          MsgSource.

  NoNH.2  Perform LSR Label No Route procedure.

            For Request No Retry

              1.   Goto NoNH.3.

            For Request Retry

              1.   Record deferred label request for FEC and Attributes
                   to be sent to MsgSource.

              2.   Start timeout. Goto NoNH.3.

  NoNH.3  DONE.
```

A.1.11. Receive Notification / Loop Detected

Summary:

When an LSR receives a Loop Detected Status Code from an LDP peer in response to a Label Request message or a Label Mapping message, it behaves as if it had received a No Route notification.

Context:

See "Receive Notification / No Route".

Algorithm:

See "Receive Notification / No Route"

Notes:

1. When the Loop Detected notification is in response to a Label Request message, it arrives in a Status Code TLV in a Notification message. When it is in response to a Label Mapping message, it arrives in a Status Code TLV in a Label Release message.

A.1.12. Receive Notification / Label Resources Available

Summary:

When an LSR receives a Label Resources Available notification from an LDP peer, it resumes sending label requests to the peer.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the Notification message.

- SAttributes. Attributes stored with postponed Label Request message.

Algorithm:

```
Res.1   Set status record indicating it is OK to send label requests
        to MsgSource.

Res.2   Iterate through Res.6 for each record of a FEC label mapping
        needed from MsgSource for which no label resources are
        available.

Res.3   Is MsgSource the next hop for FEC?
        If not, goto Res.5.

Res.4   Execute procedure Send_Label_Request (MsgSource, FEC,
        SAttributes).  If the procedure fails, terminate iteration.

Res.5   Delete record that no resources are available for a label
        mapping for FEC needed from MsgSource.

Res.6   End iteration from Res.2

Res.7   DONE.
```

A.1.13. Detect local label resources have become available

Summary:

After an LSR has sent a No Label Resources notification to an LDP peer, when label resources later become available it sends a Label Resources Available notification to each such peer.

Context:

- LSR. The LSR handling the event.

- Attributes. Attributes stored with postponed Label Mapping message.

Algorithm:

```
ResA.1  Iterate through ResA.4 for each Peer to which LSR has
        previously sent a No Label Resources notification.

ResA.2  Execute procedure Send_Notification (Peer, Label Resources
        Available)

ResA.3  Delete record that No Label Resources notification was
        previously sent to Peer.

ResA.4  End iteration from ResA.1

ResA.5  Iterate through ResA.8 for each record of a label mapping
        needed for FEC for Peer but no-label-resources.  (See Note
        1.)

ResA.6  Execute procedure Send_Label (Peer, FEC, Attributes). If the
        procedure fails, terminate iteration.

ResA.7  Clear record of FEC label mapping needed for peer but no-
        label-resources.

ResA.8  End iteration from ResA.5

ResA.9  DONE.
```

Notes:

1. Iteration ResA.5 through ResA.8 handles the situation where the LSR is using Downstream Unsolicited label distribution and was previously unable to allocate a label for a FEC.

A.1.14. LSR decides to no longer label switch a FEC

Summary:

An LSR may unilaterally decide to no longer label switch a FEC for an LDP peer. An LSR that does so must send a label withdraw message for the FEC to the peer.

Context:

- Peer. The peer.

- FEC. The FEC.

- PrevAdvLabel. The label for FEC previously advertised to Peer.

Algorithm:

```
  NoLS.1  Execute procedure Send_Label_Withdraw (Peer, FEC,
          PrevAdvLabel).  (See Note 1.)

  NoLS.2  DONE.
```

Notes:

1. The LSR may remove the label from forwarding/switching use as part of this event or as part of processing the label release from the peer in response to the label withdraw.

A.1.15. Timeout of deferred label request

Summary:

Label requests are deferred in response to No Route and Loop Detected notifications. When a deferred FEC label request for a peer times out, the LSR sends the label request.

Context:

- LSR. The LSR handling the event.

- FEC. The FEC associated with the timeout event.

- Peer. The LDP peer associated with the timeout event.

- Attributes. Attributes stored with deferred Label Request message.

Algorithm:

```
   TO.1    Retrieve the record of the deferred label request.

   TO.2    Is Peer the next hop for FEC?
           If not, goto TO.4.

   TO.3    Execute procedure Send_Label_Request (Peer, FEC).

   TO.4    DONE.
```

A.2. Common Label Distribution Procedures

This section specifies utility procedures used by the algorithms that handle label distribution events.

A.2.1. Send_Label

Summary:

The Send_Label procedure allocates a label for a FEC for an LDP peer, if possible, and sends a label mapping for the FEC to the peer. If the LSR is unable to allocate the label and if it has a pending label request from the peer, it sends the LDP peer a No Label Resources notification.

Parameters:

- Peer. The LDP peer to which the label mapping is to be sent.

- FEC. The FEC for which a label mapping is to be sent.

- Attributes. The attributes to be included with the label mapping.

Additional Context:

- LSR. The LSR executing the procedure.

- Label. The label allocated and sent to Peer.

Algorithm:

```
    SL.1   Does LSR have a label to allocate?
           If not, goto SL.9.

    SL.2   Allocate Label and bind it to the FEC.
```

```
    SL.3    Install Label for forwarding/switching use.

    SL.4    Execute procedure Send_Message (Peer, Label Mapping, FEC,
            Label, Attributes).

    SL.5    Record label mapping for FEC with Label and Attributes has
            been sent to Peer.

    SL.6    Does LSR have a record of a FEC label request from Peer
            marked as pending?
            If not, goto SL.8.

    SL.7    Delete record of pending label request for FEC from Peer.

    SL.8    Return success.

    SL.9    Does LSR have a label request for FEC from Peer marked as
            pending?
            If not, goto SL.13.

    SL.10   Execute procedure Send_Notification (Peer, No Label
            Resources).

    SL.11   Delete record of pending label request for FEC from Peer.

    SL.12   Record No Label Resources notification has been sent to Peer.
            Goto SL.14.

    SL.13   Record label mapping needed for FEC and Attributes for Peer,
            but no-label-resources. (See Note 1.)

    SL.14   Return failure.
```

Notes:

1. SL.13 handles the case of Downstream Unsolicited label distribution when the LSR is unable to allocate a label for a FEC to send to a Peer.

A.2.2. Send_Label_Request

Summary:

An LSR uses the Send_Label_Request procedure to send a request for a label for a FEC to an LDP peer if currently permitted to do so.

Parameters:

- Peer. The LDP peer to which the label request is to be sent.

- FEC. The FEC for which a label request is to be sent.

- Attributes. Attributes to be included in the label request. E.g., Hop Count, Path Vector.

Additional Context:

- LSR. The LSR executing the procedure.

Algorithm:

```
SLRq.1  Has a label request for FEC previously been sent to Peer and
        is it marked as outstanding?
        If so, Return success.  (See Note 1.)

SLRq.2  Is status record indicating it is OK to send label requests
        to Peer set?
        If not, goto SLRq.6

SLRq.3  Execute procedure Send_Message (Peer, Label Request, FEC,
        Attributes).

SLRq.4  Record label request for FEC has been sent to Peer and mark
        it as outstanding.

SLRq.5  Return success.

SLRq.6  Postpone the label request by recording label mapping for FEC
        and Attributes from Peer is needed but that no label
        resources are available.

SLRq.7  Return failure.
```

Notes:

1. If the LSR is a non-merging LSR it must distinguish between attempts to send label requests for a FEC triggered by different upstream LDP peers from duplicate requests. This procedure will not send a duplicate label request.

A.2.3. Send_Label_Withdraw

Summary:

An LSR uses the Send_Label_Withdraw procedure to withdraw a label for a FEC from an LDP peer. To do this the LSR sends a Label Withdraw message to the peer.

Parameters:

- Peer. The LDP peer to which the label withdraw is to be sent.

- FEC. The FEC for which a label is being withdrawn.

- Label. The label being withdrawn

Additional Context:

- LSR. The LSR executing the procedure.

Algorithm:

```
SWd.1  Execute procedure Send_Message (Peer, Label Withdraw, FEC,
       Label)

SWd.2  Record label withdraw for FEC has been sent to Peer and mark
       it as outstanding.
```

A.2.4. Send_Notification

Summary:

An LSR uses the Send_Notification procedure to send an LDP peer a notification message.

Parameters:

- Peer. The LDP peer to which the Notification message is to be sent.

- Status. Status code to be included in the Notification message.

Additional Context:

None.

Algorithm:

```
  SNt.1  Execute procedure Send_Message (Peer, Notification, Status)
```

A.2.5. Send_Message

Summary:

An LSR uses the Send_Message procedure to send an LDP peer an LDP message.

Parameters:

- Peer. The LDP peer to which the message is to be sent.

- Message Type. The type of message to be sent.

- Additional message contents . . . .

Additional Context:

None.

Algorithm:

This procedure is the means by which an LSR sends an LDP message of the specified type to the specified LDP peer.

A.2.6. Check_Received_Attributes

Summary:

Check the attributes received in a Label Mapping or Label Request message. If the attributes include a Hop Count or Path Vector, perform a loop detection check. If a loop is detected, cause a Loop Detected Notification message to be sent to MsgSource.

Parameters:

- MsgSource. The LDP peer that sent the message.

- MsgType. The type of message received.

- RAttributes. The attributes in the message.

Additional Context:

- LSR Id. The unique LSR Id of this LSR.

- Hop Count. The Hop Count, if any, in the received attributes.

- Path Vector. The Path Vector, if any in the received attributes.

Algorithm:

```
CRa.1   Do RAttributes include Hop Count?
        If not, goto CRa.5.

CRa.2   Does Hop Count exceed Max allowable hop count?
        If so, goto CRa.6.

CRa.3   Do RAttributes include Path Vector?
        If not, goto CRa.5.

CRa.4   Does Path Vector Include LSR Id? OR
        Does length of Path Vector exceed Max allowable length?
        If so, goto CRa.6

CRa.5   Return No Loop Detected.

CRa.6   Is MsgType LabelMapping?
        If so, goto CRa.8.  (See Note 1.)

CRa.7   Execute procedure Send_Notification (MsgSource, Loop
        Detected)

CRa.8   Return Loop Detected.

CRa.9   DONE
```

Notes:

1. When the attributes being checked were received in a Label Mapping message, the LSR sends the Loop Detected notification in a Status Code TLV in a Label Release message. (See Section "Receive Label Mapping").

A.2.7. Prepare_Label_Request_Attributes

Summary:

This procedure is used whenever a Label Request is to be sent to a Peer to compute the Hop Count and Path Vector, if any, to include in the message.

Parameters:

- Peer. The LDP peer to which the message is to be sent.

- FEC. The FEC for which a label request is to be sent.

- RAttributes. The attributes this LSR associates with the LSP for FEC.

- SAttributes. The attributes to be included in the Label Request message.

Additional Context:

- LSR Id. The unique LSR Id of this LSR.

Algorithm:

```
  PRqA.1  Is Hop Count required for this Peer (see Note 1.) ? OR
          Do RAttributes include a Hop Count? OR
          Is Loop Detection configured on LSR?
          If not, goto PRqA.14.

  PRqA.2  Is LSR ingress for FEC?
          If not, goto PRqA.6.

  PRqA.3  Include Hop Count of 1 in SAttributes.

  PRqA.4  Is Loop Detection configured on LSR?
          If not, goto PRqA.14.

  PRqA.5  Is LSR merge-capable?
```

```
              If so, goto PRqA.14.
              If not, goto PRqA.13.

      PRqA.6  Do RAttributes include a Hop Count?
              If not, goto PRqA.8.

      PRqA.7  Increment RAttributes Hop Count and copy the resulting Hop
              Count to SAttributes. (See Note 2.)
              Goto PRqA.9.

      PRqA.8  Include Hop Count of unknown (0) in SAttributes.

      PRqA.9  Is Loop Detection configured on LSR?
              If not, goto PRqA.14.

      PRqA.10 Do RAttributes have a Path Vector?
              If so, goto PRqA.12.

      PRqA.11 Is LSR merge-capable?
              If so, goto PRqA.14.
              If not, goto PRqA.13.

      PRqA.12 Add LSR Id to beginning of Path Vector from RAttributes and
              copy the resulting Path Vector into SAttributes.
              Goto PRqA.14.

      PRqA.13 Include Path Vector of length 1 containing LSR Id in
              SAttributes.

      PRqA.14 DONE.
```

Notes:

1. The link with Peer may require that Hop Count be included in Label Request messages; for example, see [ATM] and [FR].

2. For hop count arithmetic, unknown + 1 = unknown.

A.2.8. Prepare_Label_Mapping_Attributes

Summary:

This procedure is used whenever a Label Mapping is to be sent to a Peer to compute the Hop Count and Path Vector, if any, to include in the message.

Parameters:

- Peer. The LDP peer to which the message is to be sent.

- FEC. The FEC for which a label request is to be sent.

- RAttributes. The attributes this LSR associates with the LSP for FEC.

- SAttributes. The attributes to be included in the Label Mapping message.

- IsPropagating. The LSR is sending the Label Mapping message to propagate one received from the FEC next hop.

- PrevHopCount. The Hop Count, if any, this LSR associates with the LSP for the FEC.

Additional Context:

- LSR Id. The unique LSR Id of this LSR.

Algorithm:

```
PMpA.1  Is Hop Count required for this Peer (see Note 1.) ? OR
        Do RAttributes include a Hop Count? OR
        Is Loop Detection configured on LSR?
        If not, goto PMpA.21.

PMpA.2  Is LSR egress for FEC?
        If not, goto PMpA.4.

PMpA.3  Include Hop Count of 1 in SAttributes. Goto PMpA.21.

PMpA.4  Do RAttributes have a Hop Count?
        If not, goto PMpA.8.

PMpA.5  Is LSR member of edge set for an LSR domain whose LSRs do not
        perform TTL decrement AND
        Is Peer in that domain (See Note 2.).
        If not, goto PMpA.7.

PMpA.6  Include Hop Count of 1 in SAttributes. Goto PMpA.9.

PMpA.7  Increment RAttributes Hop Count and copy the resulting Hop
        Count to SAttributes. See Note 2. Goto PMpA.9.

PMpA.8  Include Hop Count of unknown (0) in SAttributes.
```

```
   PMpA.9  Is Loop Detection configured on LSR?
           If not, goto PMpA.21.

   PMpA.10 Do RAttributes have a Path Vector?
           If so, goto PMpA.19.

   PMpA.11 Is LSR propagating a received Label Mapping?
           If not, goto PMpA.20.

   PMpA.12 Does LSR support merging?
           If not, goto PMpA.14.

   PMpA.13 Has LSR previously sent a Label Mapping for FEC to Peer?
           If not, goto PMpA.20.

   PMpA.14 Do RAttributes include a Hop Count?
           If not, goto PMpA.21.

   Res.15 Is Hop Count in Rattributes unknown(0)?
           If so, goto PMpA.20.

   PMpA.16 Has LSR previously sent a Label Mapping for FEC to Peer?
           If not goto PMpA.21.

   PMpA.17 Is Hop Count in RAttributes different from PrevHopCount ?
           If not goto PMpA.21.

   PMpA.18 Is the Hop Count in RAttributes > PrevHopCount? OR
           Is PrevHopCount unknown(0)
           If not, goto PMpA.21.

   PMpA.19 Add LSR Id to beginning of Path Vector from RAttributes and
           copy the resulting Path Vector into SAttributes. Goto
           PMpA.21.

   PMpA.20 Include Path Vector of length 1 containing LSR Id in
           SAttributes.

   PMpA.21 DONE.
```

Notes:

1. The link with Peer may require that Hop Count be included in Label Mapping messages; for example, see [ATM] and [FR].

2. If the LSR is at the edge of a cloud of LSRs that do not perform TTL-decrement and it is propagating the Label Mapping message upstream into the cloud, it sets the Hop Count to 1 so that Hop Count across the cloud is calculated properly. This ensures proper TTL management for packets forwarded across the part of the LSP that passes through the cloud.

3. For hop count arithmetic, unknown + 1 = unknown.