Network Working Group Bob Thomas
Internet Draft Cisco Systems, Inc.
Expiration Date: August 2003

Alex Raj
Cisco Systems, Inc.

February 2003

LDP DoD Graceful Restart

draft-thomas-mpls-ldp-dod-restart-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

LDP graceful restart is a mechanism that helps reduce the negative effects on MPLS traffic caused by the restart of a Label Switching Router's (LSR's) control plane, specifically by the restart of its Label Distribution Protocol (LDP) component [RFC3036], on LSRs that are capable of preserving MPLS forwarding state across the restart. [RFC3478] defines procedures for LDP graceful restart for downstream unsolicited label distribution but leaves procedures for downstream on demand label distribution a subject for future study. This document defines graceful restart procedures for downstream on demand label distribution.

1. Introduction

LDP graceful restart is a mechanism that helps reduce the negative effects on MPLS traffic caused by the restart of a Label Switching Router's (LSR's) control plane, specifically by the restart of its Label Distribution Protocol (LDP) component [RFC3036], on LSRs that are capable of preserving MPLS forwarding state across the restart.

[RFC3478] defines procedures for LDP graceful restart for downstream unsolicited label distribution but leaves procedures for downstream on demand label distribution a subject for future study. This document defines graceful restart procedures for downstream on demand label distribution.

This document uses terminology introduced in [RFC3478] and, where appropriate, uses procedures specified in [RFC3478]. In addition, in places it borrows text from [RFC3478], indicating when it does so.

As in [RFC3478], for the sake of brevity:

- The phrase "the control plane" means the "the LDP component of the control plane".

- The phrase "MPLS forwarding state (for an LSP)" or simply "forwarding state (for an LSP)" means the mapping:

```
< FEC -> push outgoing_label, nexthop >
  at an LSP ingress LSR, or

< incoming_label -> swap outgoing_label, nexthop >
  at an LSP transit LSR, or

< incoming_label -> pop, nexthop >
  at an LSP transit LSR, with penultimate hop popping or
  at an LSP egress
```

The incoming_label in the forwarding state for an LSP may be a platform-wide label or an interface-specific label. Similarly, the outgoing_label learned from the nexthop may be a platform-wide label or an interface-specific label. For the procedures below that search MPLS forwarding state for an entry with a given incoming or outgoing label when the search key is an interface-specific label the match criteria includes an interface match. The procedures below also search MPLS forwarding state for an LSP or LSPs which match a given nexthop. Determining that a given LDP peer is an LSP nexthop may require comparing the nexthop with addresses advertised in Address messages from the peer.

For the sake of brevity, the phrase "LDP state (for an LSP)" means the mapping:

```
< FEC -> { outgoing_label, nexthop, nexthop neighbor LDP ID } >
  at an LSP ingress LSR, or

< FEC -> { incoming_label, upstream neighbor LDP ID },
         { outgoing_label, nexthop, nexthop neighbor LDP ID } >
  at an LSP transit LSR, or

< FEC -> { incoming_label, upstream neighbor LDP ID } >
  at an LSP egress LSR
```

The text in [RFC3478] applies here and is repeated verbatim for the reader's convenience.

> "In the case where a Label Switching Router (LSR) could preserve its MPLS forwarding state across restart of its control plane, specifically its LDP component [LDP], it is desirable not to perturb the LSPs going through that LSR (specifically, the LSPs established by LDP). In this document, we describe a mechanism, termed "LDP Graceful Restart", that allows the accomplishment of this goal".

This document extends the mechanisms defined in [RFC3478] to enable an LSR using LDP downstream on demand label distribution to accomplish the goal as well.

Continuing with the text from [RFC3478]:

> "The mechanism described ... is applicable to all LSRs, both those with the ability to preserve forwarding state during LDP restart and those without (although the latter need to implement only a subset of the mechanism described ...). Supporting (a subset of) the mechanism described here by the LSRs that can not preserve their MPLS forwarding state across the restart would not reduce the negative impact on MPLS traffic caused by their control plane restart, but it would minimize the impact if their neighbor(s) are capable of preserving the forwarding state across the restart of their control plane and implement the mechanism described here."

In specifying procedures this document distinguishes between:

- A "restarting LSR" which is one which has lost its LDP state but has retained its forwarding state; and

- A "neighbor LSR" which is one which has retained its LDP state and its forwarding state but has lost its LDP session with another LSR.

The goal of the mechanisms specified in [RFC3478] and this document is to enable uninterrupted forwarding across a control plane restart. The graceful restart mechanisms enable a restarting LSR to validate its forwarding state and recover its LDP state and a neighbor LSR to to validate its LDP state.

2. LDP Extensions

[RFC3478] specifies use of the Fault Tolerant (FT) Session TLV [LDP-FT] in the LDP Initialization message. The specified use of the FT Session TLV as described in [RFC3478] applies to the extensions for downstream on demand graceful restart as well.

This document modifies the procedures for Label Request, Label Mapping and Label Abort messages as follows for LSRs using graceful restart for downstream on demand label distribution.

- An LSR may include a Label TLV (more specifically, a Generic Label, ATM Label or Frame Relay Label TLV, as appropriate) in a Label Request message to suggest the label a downstream neighbor should use in its Label Mapping message response.

- If an LSR receives a Label Request message for which it has stale LDP state it may use the message to validate the state and it may reply with a Label Mapping message. Normal [RFC3036] procedure is to consider such a (redundant) Label Request as a protocol error and to ignore it.

- If an LSR which is not restarting receives a Label Mapping message for which it has no LDP state it responds with a Label Release message.

- If an LSR receives a Label Abort message which doesn't match a previously received Label Request message it replies with a Notification message with Label Request Aborted status code. Normal [RFC3-36] procedure is to ignore such a Label Abort.

3. Operations

The text in [RFC3478] applies here and is repeated verbatim for the reader's convenience.

"An LSR that supports functionality described in this document advertises this to its LDP neighbors by carrying the FT Session TLV in the LDP Initialization message.

"This document assumes that in certain situations, as specified in section "[Restarting] Egress LSR", in addition to the MPLS forwarding state, an LSR can also preserve its IP forwarding state across the restart. Procedures for preserving IP forwarding state across the restart are defined in [OSPF-RESTART], [ISIS-RESTART], and [BGP-RESTART]."

3.1. Procedures for restarting LSR

The text in [RFC3478] applies here and is repeated verbatim for the reader's convenience.

"After an LSR restarts its control plane, the LSR MUST check whether it was able to preserve its MPLS forwarding state from prior to the restart. If not, then the LSR sets the Recovery Time to 0 in the FT Session TLV the LSR sends to its neighbors.

"If the forwarding state has been preserved, then the LSR starts its internal timer, called MPLS Forwarding State Holding timer (the value of that timer SHOULD be configurable), and marks all the MPLS forwarding state entries as "stale". At the expiration of the timer, all the entries still marked as stale SHOULD be deleted. The value of the Recovery Time advertised in the FT Session TLV is set to the (current) value of the timer at the point in which the Initialization message carrying the FT Session TLV is sent.

"We say that an LSR is in the process of restarting when the MPLS Forwarding State Holding timer is not expired. Once the timer expires, we say that the LSR completed its restart."

The following procedures apply when an LSR using downstream on demand label distribution is in the process of restarting. The graceful restart procedures for downstream unsolicited described in [RFC3478] distinquish between the behavior of (LSP) egress and non-egress LSRs. The procedures for downstream on demand described here distinguish between the behavior of (LSP) ingress, transit and egress LSRs.

In some situations described below a restarting LSR creates LDP state which it marks as stale until it can be fully validated. When the Forwarding State Holding timer expires the restarting LSR should delete all LDP state marked as stale.

3.1.1. Restarting Ingress LSR

Reference diagram:

```
   LSP -->
   R ------ N --- ...
```

On receipt of a Label Mapping message from LSR N:

```
LabelMap(FEC, LabelTLV(outgoing_label))
```

LSR R searches its forwarding state for an LSP for FEC with the specified outgoing_label and nexthop N:

```
< FEC -> push outgoing_label, nexthop > or
< FEC -> pop, nexthop >
```

where an outgoing_label of Implicit Null matches a pop entry. If R finds no such forwarding state it is not an ingress for the FEC.

If R finds finds the forwarding state it creates LDP state for the FEC:

```
< FEC, { outgoing_label, nexthop, nexthop neighbor LDP ID } >
```

and marks it stale. Next it sends a Label Request message to N with a Label TLV suggesting outgoing_label:

```
LabelReq(MsgId_R, FEC, LabelTLV(outgoing_label))
```

On receipt of a matching Label Mapping message from N:

```
LabelMap(FEC, MsgIdTLV(MsgId_R), LabelTLV(neighbor_label))
```

R updates its LDP state for the LSP and clears the stale mark:

```
< FEC, { neighbor_label, nexthop, nexthop neighbor LDP ID } >
```

If the received neighbor_label matches outgoing_label in the forwarding state R clears the stale mark from the LSP forwarding state. Otherwise, R replaces the LSP forwarding state, as appropriate, with:

```
< FEC -> push neighbor_label, nexthop > or
< FEC -> pop, nexthop >
```

If R's Forwarding State Holding timer expires before it receives the reply to its Label Request message it deletes its forwarding state

for the LSP as per [RFC3478], clears the stale mark from its LDP state for the LSP, sends a Label Abort message to N, and follows the [RFC3036] procedures for Label Abort.

3.1.2. Restarting Egress LSR

Reference diagram:

```
       LSP -->
   --- N ------ R
```

On receipt of a Label Request message from neighbor LSR N for a FEC for which restarting LSR R is an egress and which includes a suggested label:

```
LabelReq(MsgId_N, FEC, LabelTLV(incoming_label))
```

LSR R checks whether the suggested label is compatible with its configuration for generating labels for the FEC. (An LSR may be configured to use a non-Null label, Explicit Null or Implicit Null for a FEC for which it is an egress.) If the suggested label is not compatible, R behaves as if the message had no Label TLV and treats it as specifed by [RFC3036].

If the suggested label is compatible with its configuration R attempts to locate forwarding state for the LSP using the suggested incoming_label. If incoming_label is non-Null R searches its MPLS forwarding state for an entry:

```
< incoming_label -> pop, nexthop >
```

and if incoming_label is either Explicit or Implicit Null R searches its IP forwarding state for an entry for the FEC.

If R cannot locate forwarding state corresponding to the Label Request message, it behaves as if the message had no Label TLV and treats it as specified by [RFC3036].

If R succeeds in locating forwarding state for the FEC it:

1. Updates its LDP state for the LSP:

   ```
   < FEC, { incoming_label, upstream neighbor LDP ID } >
   ```

2. Clears the stale mark from the LSP forwarding state if incoming_label is non-NULL;

3. Replies to N with a Label Mapping message:

```
LabelMap(FEC, MSGIdTLV(MsgId_N), LabelTLV(incoming_label)
```

3.1.3. Restarting Transit LSR

Reference diagram:

```
                 --- LSP -->
   ... --- Nu ------ R ------ Nd --- ...
```

Restarting LSR R must complete graceful restart procedures with upstream neighbor LSR Nu and downstream neighbor LSR Nd to recover a transit LSP. The order in which the procedures with Nu and Nd begin depends upon when the respective LDP sessions are established and the first LDP messages for the LSP are received by R.

The procedure with Nu begins with the receipt of a Label Request message with a suggested incoming_label from Nu:

```
LabelReq(MsgId_Nu, FEC, LabelTLV(incoming_label))
```

LSR R attempts to locate forwarding state for the LSP using the suggested incoming_label:

```
< incoming_label -> swap outgoing_label, nexthop >, or
< incoming_label -> pop, nexthop >
```

If R cannot locate forwarding state corresponding to the Label Request message, it behaves as if the message had no Label TLV and treats it as specified by [RFC3036].

If R succeeds in locating the forwarding state it updates its LDP state for the LSP to include incoming_label and marks it stale:

```
< FEC, { incoming_label, neighbor Nu LDP ID } >
```

At this point R's interaction with Nu for the LSP is said to be in the "waiting for downstream" state and its interaction with Nu must wait until it has a session with Nd and it completes the following procedure for clearing the stale mark on the LSP LDP state.

The procedure with nexthop LSR Nd begins with the receipt of a Label Mapping from Nd for the LSP:

```
LabelMap(FEC, LabelTLV(outgoing_label))
```

When it receives the message R searches its forwarding state for an LSR with the specified outgoing_label and nexthop Nd:

```
< incoming_label -> swap outgoing_label, nexthop >, or
< incoming_label -> pop, nexthop >
```

If R cannot locate forwarding state corresponding to the Label Mapping it ignores the message.

If R succeeds in locating the forwarding state it updates its LDP state for the LSP to include:

```
< FEC, { outgoing_label, neighbor Nd LDP ID } >
```

and marks it stale.

At this point R's interaction with Nd must wait until it has a session with Nu and the interaction with Nu for the LSP is in the "waiting for downstream" state.

When the interaction with Nu is in "waiting for downstream" state R sends Nd a Label Request message with a Label TLV suggesting outgoing_label:

```
LabelReq(MsgId_R, FEC, LabelTLV(outgoing_label))
```

On receipt of a Label Mapping message from Nd:

```
LabelMap(FEC, MsgIdTLV(MsgId_R), LabelTLV(neighbor_label))
```

that matches the Label Request message R:

1. Checks if neighbor_label matches the outgoing_label in the LSP forwarding state. If so, it clears the stale mark from the forwarding state. Otherwise, it replaces the forwarding state, as appropriate, with:

   ```
   < incoming_label -> swap neighbor_label, nexthop >, or
   < incoming_label -> pop, nexthop >
   ```

2. Updates its LDP state for the LSP and clears the stale mark:

   ```
   < FEC, { incoming_label, neighbor Nu LDP ID } >
          { neighbor_label, nexthop, neighbor Nd LDP ID } ]
   ```

At this point R may complete its interaction with Nu for the LSP by replying to Nu's Label Request message with a Label Mapping message:

  LabelMap(FEC, MsgIdTLV(MsgId_Nu), LabelTLV(incoming_label))

If R's Forwarding State Holding timer expires before it receives the reply from Nd to its Label Request message it deletes its forwarding state for the LSP as per [RFC3478], sends a Label Abort message to Nd, and deletes its LDP state for the LSP.

3.2. Restart of LDP communication with neighbor LSR

An LSP is said to be "fully established from the point of view of an LSR" if the LSR has installed forwarding state for the LSP.

When an LSR detects that an LDP downstream on demand session with a neighbor capable of preserving MPLS forwarding state across the restart has gone down, the LSR handles the LDP state learned by the session as follows:

- It marks the LDP state for LSPs that are fully established from its point of view as "stale";

- It handles the LDP state for LSPs not fully established as if the neighbor was incapable of preserving MPLS forwarding state across the restart; that is, it follows [RFC3036] procedures for session loss for these LSPs.

The text in [RFC3478] regarding neighbor LSR behavior when an LDP session with a neighbor is lost and later re-established applies here and is repeated verbatim for the reader's convenience.

"After detecting that the LDP session with the neighbor went down, the LSR tries to re-establish LDP communication with the neighbor following the usual LDP procedures.

"The amount of time the LSR keeps its stale label-FEC bindings is set to the lesser of the FT Reconnect Timeout, as was advertised by the neighbor, and a local timer, called the Neighbor Liveness Timer. If within that time the LSR still does not establish an LDP session with the neighbor, all the stale bindings SHOULD be deleted. The Neighbor Liveness Timer is started when the LSR detects that its LDP session with the neighbor went down. The value of the Neighbor Liveness timer SHOULD be configurable.

"If the LSR re-establishes an LDP session with the neighbor within the lesser of the FT Reconnect Timeout and the Neighbor Liveness Timer, and the LSR determines that the neighbor was not able to preserve its MPLS forwarding state, the LSR SHOULD immediately delete all the stale label-FEC bindings received from that

neighbor. If the LSR determines that the neighbor was able to preserve its MPLS forwarding state (as was indicated by the non-zero Recovery Time advertised by the neighbor), the LSR SHOULD further keep the stale label-FEC bindings, received from the neighbor, for as long as the lesser of the Recovery Time advertised by the neighbor, and a local configurable value, called Maximum Recovery Time, allows."

When the LSR determines that the neighbor has preserved its MPLS forwarding state it starts an internal timer called the LDP Recovery timer.

Other than to specify the procedure when the Neighbor Liveness Timer expires [RFC3478] need not address LSR behavior following session loss prior to re-establishment. The procedures for downstream on demand specify neighbor LSR behavior for individual LSPs during this period as well as following session re-establishment.

3.2.1. Neighbor Ingress LSR

Reference Diagram:

```
LSP -->
N ------ P --- ...
```

Prior to session re-establishment if the nexthop for an LSP FEC changes from peer LSR P LSR N clears the stale mark from the LSP LDP state and follows the [RFC3036] procedures for a nexthop change with the following difference. Where the procedures require N to send a Label Release message to Pd N omits the message.

Following session establishment with downstream peer LSR P, neighbor LSR N scans its LDP state for LSPs for which it is the ingress LSR and which use P as nexthop:

```
< FEC, { outgoing_label, nexthop, nexthop neighbor LDP ID } >
```

For each such LSP N sends P a Label Request message with a Label TLV suggesting outgoing_label:

```
LabelReq(MsgId_N, FEC, LabelTLV(outgoing label))
```

On receipt of a matching Label Mapping message from P:

```
LabelMap(FEC, MsgIdTLV(MsgId_N), LabelTLV(neighbor_label))
```

N updates its LDP state for the LSP and clears the stale mark:

< FEC, { neighbor_label, nexthop, nexthop neighbor LDP ID } >

If the received neighbor_label matches outgoing_label in the LSP forwarding state N takes no further action; otherwise, it replaces the LSP forwarding state, as appropriate, with:

< FEC -> push neighbor_label, nexthop > or
< FEC -> pop, nexthop >

If N's LDP Recovery timer expires before it receives the reply to its Label Request message it deletes its forwarding state for the LSP, removes the outgoing_label from the LDP state for the LSP, clears the stale mark from the LDP state, sends a Label Abort message to P, and follows the [RFC3036] procedures for Label Abort.

3.2.2. Neighbor Egress LSR

Reference Diagram:

```
            LSP -->
    ... --- P ------ N
```

Prior to session re-establishment if the nexthop for the LSP FEC (i.e., the FEC route) is lost N follows the [RFC3036] procedures for a change in FEC nexthop with the following difference. Where the procedures require N send a Label Withdraw message to upstream peer LSR Pu it omits the message and behaves as if it had sent the message and Pu had replied with a Label Release message.

Following session establishment with upstream LSR P neighbor LSR N scans its LDP state for LSPs for which it is an egress and P is the previous hop:

[FEC, { incoming label, upstream neighbor LDP ID }]

For each such LSP it sends a Label Mapping to P:

LabelMap(FEC, LabelTLV(incoming label)

On receipt of a Label Request message with a suggested label from P:

LabelReq(MsgId_P, FEC, LabelTLV(incoming_label);

neighbor LSR N attempts to locate LDP state using the FEC and suggested incoming_label:

If N cannot locate the LDP state message, it behaves as if the

message had no Label TLV and treats it as specified by [RFC3036].

If N succeeds in locating the LDP state, it clears the stale mark from the LDP state and replies to P with a Label Mapping message:

```
LabelMap(FEC, MsgIdTLV(MsgId_P), LabelTLV(incoming label)
```

If N's LDP Recovery timer expires before it receives a Label Request message from P for the LSP it deletes its LDP state for the LSP and any MPLS forwarding state for the LSP.

3.2.3. Neighbor Transit LSR

There are 3 cases for a neighbor LSR acting as a transit LSR.

Case 1: LDP session with upstream peer LSR Pu is lost and recovered:

Reference Diagram:

```
                  -- LSP -->
   ... --- Pu ------ N --- ...
```

Prior to session re-establishment the following could occur for an LSP for which upstream LSR Pu is the previous hop:

- N receives a Label Withdraw message from the LSP nexthop LSR (Pd). In this case N follows the procedures specified in [RFC3036] for receipt of a Label Withdraw with the following differences. Regardless of the control method used to establish the LSP N behaves as if it had used ordered control. Where the procedures require N send a Label Withdraw message to upstream peer LSR Pu it omits the message and behaves as if it had sent the message and Pu had replied with a Label Release message.

- The nexthop for the LSP FEC changes. In this case N follows the [RFC3036] procedures for a change in FEC nexthop with the following difference. Where the procedures require N send a Label Withdraw message to upstream peer LSR Pu it omits the message and behaves as if it had sent the message and Pu had replied with a Label Release message.

Following session establishment with upstream peer LSR Pu neighbor LSR N scans its LDP state for LSPs for which LSR Pu is the previous hop. For each such LSP N behaves as described in section "Neighbor Egress LSR" with the following difference.

If N's LDP Recovery timer expires before it receives a Label Request message from Pu for the LSP it clears the stale mark from its LDP state for the LSP and behaves as if it had received a Label Release message from Pu and follows the [RFC3036] Label Release procedures.

Case 2: LDP session with downstream peer LR Pd is lost and recovered:

Reference Diagram:

```
        --- LSP -->
    ... --- N ------ Pd --- ...
```

Prior to session re-establishment the following could occur for an LSP for which downstream LSR Pd is the nexthop:

- The LSP previous hop LSR (Pu) releases its label for the LSP. In this case neighbor LSR N follows the [RFC3036] procedures for Label Release with the following difference. Where the procedures require N to send a Label Release message to Pd N omits the message.

- The nexthop for an LSP FEC changes. In this case N clears the stale mark from the LSP LDP state and follows the [RFC3036] procedures for a nexthop change with the following difference. Where the procedures require N to send a Label Release message to Pd N omits the message.

Following session establishment with downstream peer LSR Pd, neighbor LSR N scans its LDP state for LSPs for which Pd is nexthop. For each such LSP N behaves as described in section "Neighbor Ingress LSR" with the following difference.

If N's LDP Recovery timer expires before it receives the reply to its Label Request message it behaves as follows:

1. It deletes its forwarding state for the LSP and clears the stale mark from its LDP state for the LSP.

2. It sends a Label Abort message to Pd and follows the [RFC3036] procedures for Label Abort.

3. With respect to its upstream neighbors it behaves as if it had received a Label Withdraw message from Pd and sends Label Withdraws to some or all of its upstream neighbors as specified by [RFC3036].

Case 3: LDP sessions with upstream peer LSR Pu and downstream peer LSR Pd are lost and recovered.

Reference Diagram:

```
                ---> LSP -->
   ... --- Pu ------ N ------ Pd --- ...
```

Neighbor LSR N must complete graceful restart procedures with upstream peer LSR Pu and downstream peer LSR Pd to recover a transit LSP. The order in which the procedures with Pu and Pd begin depends upon when the respective LDP sessions are established and the first LDP messages for the LSP are received by N.

Prior to session re-establishment with Pd if the nexthop for an LSP FEC changes from downstream peer LSR Pd, neighbor LSR N follows the [RFC3036] procedures for a change in FEC nexthop with the following differences. If the session with Pu has not been re-established where the [RFC3036] procedures require N send a Label Withdraw message to upstream peer LSR Pu N omits the message and behaves as if it had sent the message and Pu had replied with a Label Release message; and, where the procedures require N to send a Label Release message to Pd N omits the message.

Following session establishment with upstream LSR Pu neighbor LSR N scans its LDP state for LSPs for which Pu is the previous hop:

```
< FEC, { incoming label, upstream neighbor LDP ID } >
```

For each such LSP it sends a Label Mapping to Pu:

```
LabelMap(FEC, LabelTLV(incoming label)
```

On receipt of a Label Request message with a suggested label from Pu:

```
LabelReq(MsgId_Pu, FEC, LabelTLV(incoming_label);
```

N attempts to locate LDP state using the FEC and suggested incoming_label:

If N cannot locate the LDP state message, it behaves as if the message had no Label TLV and treats it as specified by [RFC3036].

If N succeeds in locating the LDP state its interaction with Pu for the LSP is said to be in the "waiting for downstream" state and its interaction with Pu for the LSP must wait until it has a session with Pd and it completes the following procedure for clearing the

stale mark on the LSP LDP state.

Following session establishment with downstream peer LSR Pd, neighbor LSR N scans its LDP state for LSPs which use Pd as nexthop:

```
  < FEC, { ... } >
         { outgoing_label, nexthop, nexthop neighbor LDP ID } >
```

Each such LSP the interaction with Pd is considered to be in "waiting for upstream" state. At this point for a given LSP N must wait until the interaction with Pu is in "waiting for downstream" state.

When the interaction with Pu is in "waiting for downstream" state and the interaction with Pd is in "waiting for upstream" state N sends Pd a Label Request message with a Label TLV suggesting outgoing_label:

```
  LabelReq(MsgId_N, FEC, LabelTLV(outgoing label))
```

On receipt of a Label Mapping message from Pd:

```
  LabelMap(FEC, MsgIdTLV(MsgId_N), LabelTLV(neighbor_label))
```

that matches the Label Request message N:

1. Checks if the received neighbor_label from Pd matches the outgoing_label in the LSP forwarding state. If not N replaces the LSP forwarding state with:

   ```
   <incoming_label -> neighbor_label, nexthop>
   ```

2. Updates its LDP state for the LSP and clears the stale mark:

   ```
   < FEC, { neighbor_label, nexthop, nexthop Pd LDP ID } >
   ```

At this point, N may complete its interaction with Nu for the LSP by replying to Pu's Label Request message with a Label Mapping message:

```
  LabelMap(FEC, MsgIdTLV(MsgId_Pu), LabelTLV(incoming label)
```

If N's LDP Recovery timer expires before it receives the reply from Pd to its Label Request message it deletes its forwarding state for the LSP as per [RFC3478], sends a Label Abort message to Pd, and deletes its LDP state for the LSP.

3.3. Non-Merging LSRs

The forwarding state for a given FEC with N LSPs on a non-merging restarting LSR is:

```
[ incoming_label_1 -> swap outgoing_label_1, nexthop ]
[ incoming_label_2 -> swap outgoing_label_2, nexthop ]
...
[ incoming_label_N -> swap outgoing_label_N, nexthop ]
```

In some situations it may be important for a restarting transit LSR to preserve the existing LSPs for a given FEC; that is, to ensure that following completion of graceful restart each incoming label is (re)connected to the same outgoing label for each LSP.

The procedures described in section "Restarting Transit LSR" ensure that the LSR recovers each LSP for the FEC correctly. Label Request messages from the upstream neighbor LSR identify the FEC LSPs by incoming_label. This enables the LSR to determine the corresponding outgoing_label for the LSP from the forwarding state. Similarly Label Mapping messages from the downstream neighbor LSR identify the FEC LSPs by outgoing_label. This enables the LSR to determining the incoming_label for the LSP from the forwarding state.

3.4. Mixed DU / DoD operation

An LSR may use downstream unsolicited label distribution with some neighbors and downstream on demand with others. A situation of interest is where a restarting Transit LSR uses different methods with its upstream and downstream neighbors for a given LSP.

The discussion below assumes all labels are Generic Labels. In situations where the label types are different (e.g., the downstream on demand session distributes ATM Labels and the downstream unsolicited session distributes Generic Labels) the forwarding state and the way LSR R manages it are implementation dependent and are likely to be be different in detail, but are similar in kind, to situations where all labels are Generic.

For this situation the LSP LDP state to be recovered is:

```
< FEC, { incoming_label, neighbor Nu LDP ID } >
       { outgoing_label, nexthop, neighbor Nd LDP ID } >
```

and the LSP forwarding state to be validated is:

```
< incoming_label -> outgoing_label, nexthop >, or
< incoming_label -> pop, nexthop >
```

To fully recover the LSP R must establish sessions with Nu and Nd and complete restart procedures with each. In general R will not know until a session is established whether it is downstream on demand or downstream unsolicited.

There are two cases to consider:

Case 1: Upstream session is downstream on demand;
        downstream session is downstream unsolicited.

```
                  --- LSP -->
     ... --- Nu ------ R ------ Nd --- ...
                DoD       DU
```

The graceful restart procedures for restarting LSR R's session with Nu are those described above in section "Restarting Egress LSR" without the configuration check. Successful completion of the procedures with Nu results in the recovery of part of the LSP LDP state:

```
< FEC, { incoming_label, upstream neighbor  LDP ID } >
```

and validation of the incoming_label part of the forwarding state.

Completion of the [RFC3478] procedures with Nd results in recovery of part of the LDP state for the LSP:

```
< FEC, { incoming_label },
       { outgoing_label, downstream neighbor LDP ID } >
```

and full validation of the LSP LDP state.

Case 2: Upstream session is downstream unsolicited;
        downstream session is downstream on demand.

```
                  --- LSP -->
     ... --- Nu ------ R ------ Nd --- ...
                 DU      DoD
```

Following establishment of its session with DU upstream neighbor Nu, restarting LSR R considers all LSPs that transit Nu-R-Nd (which it has yet to identify from it preserved forwarding state and interactions with Nd) to be in "waiting for downstream" state, and it follows the [RFC3478] procedures for its session with Nu.

After establishment of R's session with DoD downstream neighbor Nd, on reciept of a Label Mapping message from Nd:

```
LabelMap(FEC, LabelTLV(outgoing_label))
```

LSR R searches its forwarding state for an LSR with the specified outgoing_label and nexthop Nd:

```
< incoming_label -> swap outgoing_label, nexthop >, or
< incoming_label -> pop, nexthop >
```

If R cannot locate forwarding state corresponding to the Label Mapping it ignores the message. If R succeeds in locating the forwarding state it updates its LDP state for the LSP to include:

```
< FEC, { incoming_label, ... },
       { outgoing_label, neighbor Nd LDP ID } >
```

At this point R's interaction with Nd must wait until it has a session with Nu and the interaction with Nu for the LSP is in the "waiting for downstream" state.

When the interaction with Nu for the LSP is in "waiting for downstream" state R:

1. Sends Nd a Label Request message with a Label TLV suggesting outgoing_label:

   ```
   LabelReq(MsgId_R, FEC, LabelTLV(outgoing_label))
   ```

2. On receipt of a Label Mapping message from Nd:

   ```
   LabelMap(FEC, MsgIdTLV(MsgId_R), LabelTLV(neighbor_label))
   ```

   that matches the Label Request message R sends Nu a Label Mapping message:

   ```
   LabelMap(FEC, LabelTLV(incoming_label))
   ```

3. Clears the stale mark from the LSP forwarding state.

3.5. LSP attributes and loop detection

[RFC3036] procedures for handling Hop Count and Path Vector TLVs in Label Mapping and Label Request messages apply following LDP session re-establishment during the period while Forwarding State and LDP Recovery timers are running. This includes the procedures for the optional loop detection mechanism specified for use with downstrean on demand label distribution.

4. Security Considerations

This document introduces no security issues beyond those previously identified by [RFC3036] and [RFC3478].

5. Acknowledgements

The authors would like to thank Bin Mo, Bibhuti Kar, Iftekhar Hussain, Richard Lam and Bob Fish for their input.

6. References

[BGP-RESTART] "Graceful Restart Mechanism for BGP", draft-ietf-idr-restart-03.txt, work in progress.

[ISIS-RESTART] "Restart signaling for ISIS", draft-ietf-isis-restart-01.txt, work in progress.

[LDP-FT] Farrell A., et al, "Fault Tolerance for the Label Distribution Protocol (LDP)", draft-ietf-mpls-ldp-ft-06.txt, September 2002, work in progress.

[OSPF-RESTART] "Hitless OSPF Restart", draft-ietf-ospf-hitless-restart-01.txt, work in progress.

[RFC3036] Andersson, L., et al, "LDP Specification, RFC3036", January 2001.

[RFC3478] Leelanivas, M., et al, "Graceful Restart Mechanism for LDP", draft-ietf-mpls-ldp-restart-06.txt, October 2002, work in progress.

7. Authors' Addresses

Alex Raj
Cisco Systems, Inc
300 Apollo Dr.
Chelmsford, MA 01824
email: araj@cisco.com

Bob Thomas
Cisco Systems, Inc.
300 Apollo Dr.
Chelmsford, MA 01824
email: rhthomas@cisco.com