Available online at www.sciencedirect.com
SCIENCE DIRECT®
Optical Switching and Networking 2 (2005) 148–162
Optical Switching and Networking
www.elsevier.com/locate/osn

Recovery from control plane failures in the LDP signalling protocol

Jing Wu[a,*], Michel Savoie[a], Hussein Mouftah[b]

[a]*Communications Research Centre Canada, 3701 Carling Avenue, Ottawa, Ontario, Canada K2H 8S2*
[b]*School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*

Received 11 September 2004; received in revised form 13 July 2005; accepted 26 July 2005
Available online 29 August 2005

Abstract

The Label Distribution Protocol (LDP) needs to recover its state information after a control plane failure, so that the established connections in the data plane are not disrupted by any new connection set-up. We propose a backup mechanism to store the LDP state information in an upstream neighbour node. The backup LDP state information is synchronized with the original LDP state information in a downstream node when the LDP sets up or tears down connections. Then, we propose a two-step LDP state information recovery, which uses a fast LDP state information recovery to recover what labels are idle before a control plane failure, and a detailed LDP state information recovery to fully recover all LDP state information. The fast LDP state information recovery is realized as part of the LDP initialization, allowing a restarting LDP session to process new connection set-up requests as soon as possible, without interfering existing connections. The detailed LDP state information recovery performs in the background in parallel to the normal LDP operations. When an LDP connection teardown requires the LDP state information that has not yet been recovered, an on-demand query based LDP state information recovery is conducted. The performance analysis demonstrates that our proposal achieves fast LDP recovery for the core label state information. It features scalable LDP state information storage and recovery by only involving a pair of neighbour nodes.

Keywords: Label Distribution Protocol; Signalling protocol; Multi-Protocol Label Switching; Control plane; Failure recovery

1. Introduction

The reliability of the control plane of a Multi-Protocol Label Switching (MPLS) network is important [1]. New generation MPLS switches/routers are able to maintain label switching state in the data plane in certain control plane failures or during its maintenance [2]. In a Generalized Multi-Protocol Label Switching (GMPLS) network, the integrity of the control and data planes is more or less independent when they are physically separate [3–5]. Therefore, it is required to minimize service interruptions in a control plane failure or during its maintenance.

* Corresponding author. Fax: +1 613 990 8382.
E-mail addresses: jing.wu@crc.ca (J. Wu), michel.savoie@crc.ca (M. Savoie), mouftah@site.uottawa.ca (H. Mouftah).

1573-4277/$ - see front matter
doi:10.1016/j.osn.2005.07.001

The Label Distribution Protocol (LDP) is a signalling protocol, which is used to set up, maintain and tear down connections in an MPLS network [6]. The Constraint-based Routing Label Distribution Protocol (CR-LDP) is an extension of the LDP, and is used as a signalling protocol for GMPLS-controlled circuit-switched networks [7]. Between two adjacent control nodes, an LDP session is used to exchange LDP messages and control the corresponding data plane links. A failed LDP session results in the loss of LDP state information. The lost state information cannot be automatically recovered in a new restarting LDP session.

Our approach maintains a backup copy of the LDP state information at the upstream neighbour of each node. The backup copy is synchronized with the original copy when exchanging regular LDP messages. When initializing an LDP session, the LDP session attempts to recover some basic state information from the previous LDP session. Unlike the fault tolerance for the LDP [8], our approach aims at recovering the LDP state information not only in a signalling channel failure, but also in a control node failure or during its maintenance. Our approach is able to recover from a downstream control node failure, which cannot be achieved in the graceful restart mechanism for the LDP [2]. Our approach handles all control plane failures in a unified manner so that the control plane failure type does not have to be diagnosed. Assuming no adjacent control nodes fail at the same time, our approach can recover the LDP state information in any single-point failure. Compared to querying neighbour control nodes to recover the LDP state information after initialization [9], our approach has an advantage that the new LDP session enters the operational state faster. The LDP queries can be used to recover more accurate LDP state information after a coarse and fast recovery of our approach. Compared to the previously proposed LDP recovery [10], our approach's performance is significantly improved by taking advantage of a two-step recovery and parallelism in the transfer and process of the LDP state information.

This paper is organized as the following: in Section 2, we will discuss our design motivations and existing techniques for LDP recovery. Then, we will briefly present an overview of the LDP operations in Section 3 and the maintenance of backup label information in the LDP in Section 4. A distributed LDP recovery method is proposed in Section 5. The proposed failure recovery technique is applied to an example in Section 6. In Section 7, we analyze the performance of the proposed LDP recovery mechanism. We conclude the paper in Section 8.

2. Motivations and existing techniques

The LDP is vulnerable to hardware and software failures [11–13]. This is in contrast to the fault tolerance of the resource reservation protocol (RSVP), which uses periodical state refreshments [11]. Routing protocols such as the Open Shortest Path First (OSPF) or the Intermediate System to Intermediate System (IS–IS) are fairly fault tolerant. They exchange information through periodical link state advertisements [3,14]. If a control plane failure happens, they can still recover after the fault is fixed and the link state advertisement resumes. The LDP's difficulty in failure recovery is inherent to hard-state protocols, e.g., the Border Gateway Protocol (BGP) [15] and the Private Network to Network Interface (PNNI) [16], because their status information is not automatically refreshed.

The importance of handling control plane failures and recovery for a signalling protocol was identified in [1,17]. It was suggested that any control plane failure must not result in releasing established calls and connections. Upon recovery from a control plane failure, the recovered node must have the ability to recover the status of the calls and connections established before the failure. Calls and connections in the process of being established (i.e. pending call/connection set-up requests) should be released or continued with set-up.

Several techniques have been proposed for the LDP failure recovery. In addition, generic failure recovery techniques for distributed systems or control systems may also be applied to the LDP failure recovery. They have different assumptions and objectives, resulting in different recovery capability, recovery accuracy and speed, and different implementation overhead and cost. These techniques include the following:

1. Redundant control node hardware or LDP signalling software. A standby backup control node or LDP signalling module may replace a failed one in real time;

2. Persistent storage of relevant information. After a reboot, such a control node may maintain the LDP state information, configuration information, and control plane neighbour information;
3. Backup signalling channels [4,5,18]. The LDP messages will be re-routed over the backup signalling channels if the primary signalling channel fails;
4. Message logging [19]. All LDP messages are securely stored and replayed if a failure occurs;
5. Fault tolerant mechanism for the LDP [8]. The unacknowledged LDP messages are re-sent and the LDP session status is re-synchronized between upstream and downstream nodes;
6. Graceful restart mechanism for the LDP [2,20]. A downstream node notifies its upstream neighbour the label mapping information that the downstream node preserves through a restart;
7. Control plane inquires the data plane about the channel status. Depending on the data plane capability, the channel status, e.g., in-use or idle, may be inquired to recover a control node's lost status information;
8. Query-and-reply based LDP state information recovery [9]. This method can recover detailed LDP state information and is not limited to only recover from the backup state information at direct neighbours;
9. Management system centralized recovery. The network management system may conduct complicated coordination and information transfers, but in a less real time manner.

Our first goal is to provide a unified LDP failure recovery approach that handles any single-point failure. This requires that a control node's failure recovery should not rely on the state information stored in the node itself. Although locally preserved state information certainly helps our approach to be more efficient, our approach does not rely on it. However, it is reasonable to assume a control node is able to recover or discover the LDP neighbour information and configuration information. In particular, as part of the configuration information, the label space information should be recovered either by the control node itself or through the management system. In MPLS, the label space defines the boundary of valid labels. In GMPLS, the label space enumerates the configured data plane channels. The recovery from multiple simultaneous failures is not of primary interest of our approach, since they can be handled by existing mechanisms for failure recovery, such as the fault management function in the network management system. Examples of such multiple failures include a failure interfering with an ongoing failure recovery, simultaneous failure of a data link and its associated control node, etc. The management system monitors all failures and should take proper actions, e.g., disable the automatic LDP failure recovery if such recovery is impossible or undesirable.

Our second goal is to provide a fast recovery of the basic LDP state information. A complete recovery of all LDP state information takes time to finish. The recovery time depends on the number of data plane connections affected by the failed LDP session and the details of the LDP state information. Our approach is to accomplish a complete recovery in two steps: (i) a fast and coarse recovery that only recovers the LDP state information about which labels (or channels) in the data plane are idle before the failure; (ii) a detailed recovery that recovers the complete LDP state information. One advantage of this two-step approach is that the new established LDP session enters the operational state faster so that new connection set-up or teardown requests can be handled in a timely fashion. It is safe to use idle labels (or channels) in the data plane to establish new connections and not interrupt established connections. Another advantage is that a detailed LDP state information recovery (likely based on the query-and-reply approach) can be conducted in parallel with the normal LDP operation. If a given LDP operation requires the LDP state information that has not yet been completely recovered, an on-demand query can be issued immediately to collect the particular information.

Our last goal is to avoid additional hardware and requirements being imposed on the data plane equipment. Apparently, a standby backup control node can effectively recover the LDP state information. Queries to the data plane about its label (or channel) status may also achieve an effective LDP recovery. Our approach compliments these two approaches without additional requirements being imposed on a backup control node and data plane equipment. Our approach should be used only when these two approaches are unavailable. A control node

should notify the management system of inability to recover certain LDP state information, e.g., inability to synchronize connection states, or simultaneous failures of two adjacent control nodes. Then the management system should take further actions.

3. Overview of the LDP operations

The LDP is responsible for exchanging the signalling messages between nodes to control the connections between the corresponding data plane equipment. Each pair of adjacent nodes runs an LDP session to exchange messages. Each side of an LDP session uses an LDP entity, which is a software process together with a set of state variables and timers. Only when an upstream and a downstream LDP entity are well synchronized through a proper LDP initialization, the signalling messages to set up or tear down connections can be handled correctly. The LDP messages are organized into four categories [6]: 1. *Discovery messages*, used to announce and maintain the presence of a node; 2. *Session messages*, used to establish, maintain, and terminate an LDP session; 3. *Advertisement messages*, used to create, change and delete label mappings, thus set up and tear down connections in the data plane; 4. *Notification messages*, used to provide advisory information and to signal error information. An LDP session can be described by a state machine [6].

The LDP operations are illustrated in Fig. 1. An LDP-speaking node indicates its presence in a network by sending Hello messages periodically. When a node chooses to establish an LDP session with another node that is learned via the Hello message, an LDP session will be initialized between the two nodes. Upon successful completion of the initialization procedure, the two nodes become LDP peers, and may exchange advertisement messages to set up or tear down connections in the data plane. The status of the connections in the data plane is represented as the label status in the LDP. The LDP advertisement and notification messages are transported over TCP to ensure a reliable and orderly delivery of the messages.

4. Maintaining a backup of the label state information

In the LDP, a downstream node maintains the label state information. The label state information

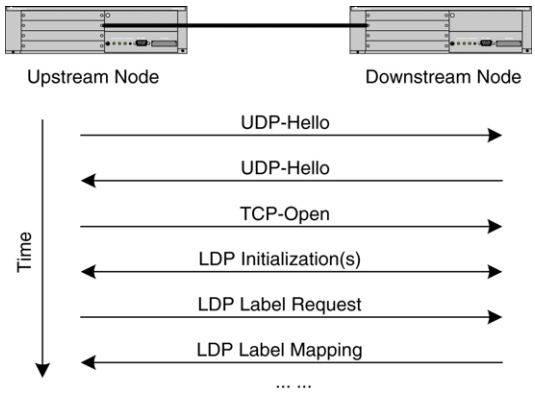

Fig. 1. LDP operations.

management in the standard LDP is shown in Fig. 2. In a connection set-up, the label state information is updated when a downstream node assigns a label to the connection. To set up a connection in the data plane, an upstream node (with respect to the direction of the connection) explicitly requests a label from the downstream node in an LDP Label Request message. The downstream node then retrieves the information about the available channels (labels) for that incoming link. If a channel is available and the policy allows it, the downstream node reserves a channel and assigns a label. By assigning a label to the connection, the label status is changed from idle to in-use. At the same time, the associated connection information is stored in the Label Information Database (LID) in the downstream node. Such connection information is specific to the data plane technology. In response to the LDP Label Request message, the downstream node sends back an LDP Label Mapping message to the upstream node. After the upstream node receives the LDP Label Mapping message, it can start using the connection corresponding to the indicated label [7].

In a connection teardown, the label state information is updated in a downstream node when the downstream node receives a teardown confirmation from the upstream node. A connection teardown can be initiated by the ingress node (i.e., the source end) or the egress node (i.e., the destination end). In the egress-initiated teardown, the egress node sends an LDP Label Withdraw message to its upstream peer node. If the upstream node decides to tear down that connection, it sends back an LDP Label Release message

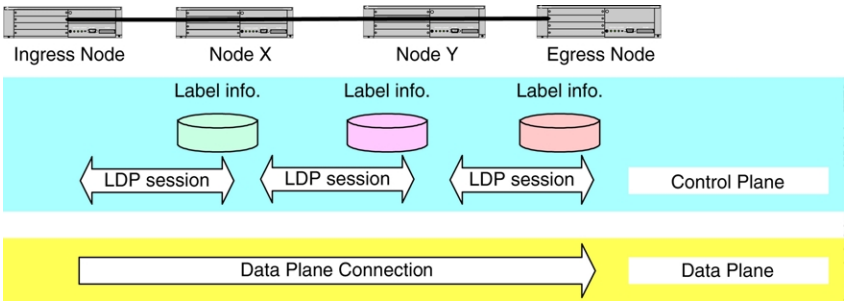

Fig. 2. A downstream node manages the link status information in the LDP.

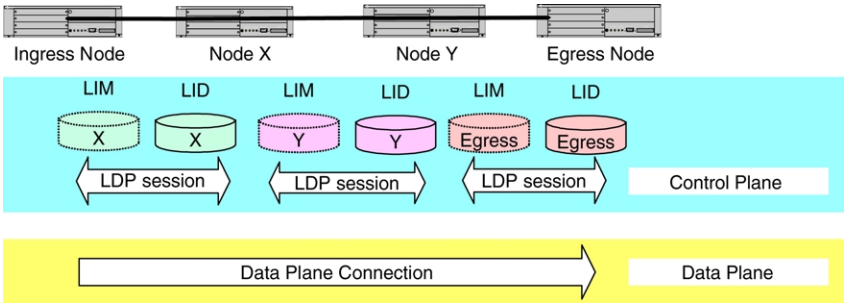

Fig. 3. Redundant storage of the label state information.

and stops using that connection. Upon receiving that LDP Label Release message, the egress node updates the label state to idle and stops using that connection. Each LDP session repeats this procedure in the opposite direction of the connection. When the ingress node wants to tear down an established connection, it first sends an LDP notification message to the egress node so that no loss of signal error will be triggered at the egress node. Then the same procedure used in the egress-initiated teardown is applied.

We propose to introduce a Label Information Mirror (LIM) in an upstream node (Fig. 3). Each LIM is a copy of the LID in the corresponding downstream node. Because labels only have local significance with respect to the link they refer to, both a LID and a LIM only store the information about labels regarding a specific link and have no global significance. This makes the recovery mechanism scalable and enables it to be deployed on a per LDP session basis. As we assume no adjacent control nodes fail at the same time, either a LID or its corresponding LIM will always be accessible.

The label state information includes the status of each configured label in the label space. A valid label has one of the following statuses: idle (i.e., free), in-use, or reserved (i.e., a transient state after receiving a label request message and before replying a label mapping message; sometimes this state is called “pending”). The label space represents the configured channels in the data plane and is part of the configuration information. Together with the label status, a LID or LIM also stores additional label-related information, which is specific to the data plane technology. For example, in MPLS, such label-related information includes the identifier of a Forwarding Equivalence Class (FEC), the label operation at a downstream Label Switched Router (LSR) (e.g., label swapping, label stack push/pop operation), the link layer format for an incoming link, etc. [21, 22]. In a GMPLS-controlled Wavelength Division Multiplexing (WDM) network, such label-related information includes the ingress and egress nodes of a lightpath, protection scheme, the wavelength operation at a downstream node (e.g., add/drop

operation, wavelength conversion, optical to electrical conversion), etc. [23].

5. A distributed LDP recovery method

Our proposed LDP recovery method includes a synchronization procedure for the label state information redundantly stored in upstream and downstream nodes, and a two-step recovery procedure: (i) a fast and coarse LDP state information recovery during the initialization of a restarting LDP session; and (ii) a detailed LDP state information recovery running in parallel to the normal LDP operations.

5.1. Synchronization procedure of a label information mirror and a label information database

A LIM and its corresponding LID are synchronized by carrying additional label-related information in a LID to an upstream node. During the "cold" initialization phase of an LDP session (i.e., initialization from scratch), a LIM and its corresponding LID are initialized based on the actual channel configuration in the data plane. As a result, the LIM and its corresponding LID initially have identical contents, where all configured labels have the idle state. When a node requests a label, it follows the standard LDP procedure. In addition to the regular procedure, additional label-related information in the LID is carried in an LDP Label Mapping message and an upstream node updates its LIM when it receives the message. So both the LIM and the LID are synchronized after a data plane connection is established, i.e. their contents are identical. Besides the regular procedure in the connection termination phase, an upstream node updates its LIM when it sends an LDP Label Release message to a downstream node. In this way, both the LIM and the LID are synchronized after terminating a data plane connection.

The synchronization of a LIM and LID after a control plane failure that affects the LDP session is part of the proposed two-step LDP recovery, which will be discussed shortly.

5.2. A fast LDP state information recovery

We propose a fast LDP state information recovery to recover a control node's information about what labels are idle, so that new connection set-ups do not disrupt the established connections and the LDP signalling can process new connection set-up requests sooner.

An upstream node and a downstream node exchange information about the idle labels for the LDP session between both nodes in the LDP session initialization. Four new Type-Length-Value objects (TLVs) are defined for the LDP session initialization message: LIM, LID, Recovery (Rcvy) and Cork TLVs. An upstream node uses the LIM TLV to notify its downstream peer node about the idle labels in a LIM. A downstream node uses the LID TLV to notify its upstream node about the idle labels in a LID. The idle labels can be enumerated as individual labels, or as groups of consecutive labels by specifying the boundaries of the label groups, or as a combination of the former two types. The Rcvy TLV is a flag to indicate to a node's LDP peer whether the node intends to conduct the proposed LDP recovery. By default, an LDP session initialization message should include the Rcvy flag. However, the management system or an operator can overwrite such default by excluding the Rcvy TLV from the LDP initialization message, so that the operation of the proposed LDP recovery is disabled. Examples of such occasions include the "cold" LDP initialization where the LDP recovery is unnecessary, simultaneous failure of two adjacent nodes where the label state information is completely lost. The Cork TLV is a flag to indicate the end of sending a complete list of idle labels in a LIM or LID, since the transfer of a complete list can be split into multiple LIM or LID TLVs.

The fast LDP state information recovery procedure is an extension of the standard LDP initialization procedure. The state machine specification is given in Fig. 4. The following notations are used in the state machine specification. A rectangular block represents a major state. An arrow from a state to another represents a state transition, where the condition of the transition is specified as the first half notation beside the arrow (before a slash mark "/") and the actions after the transition are specified as the second half notation. An elliptical block is a sub-state within a major state. A hexagonal block within an elliptical block represents a micro-state. In the diagram, "Rx" denotes "receive", "Tx" denotes "transmit", "msg" denotes "message", and "init" denotes "initialization".

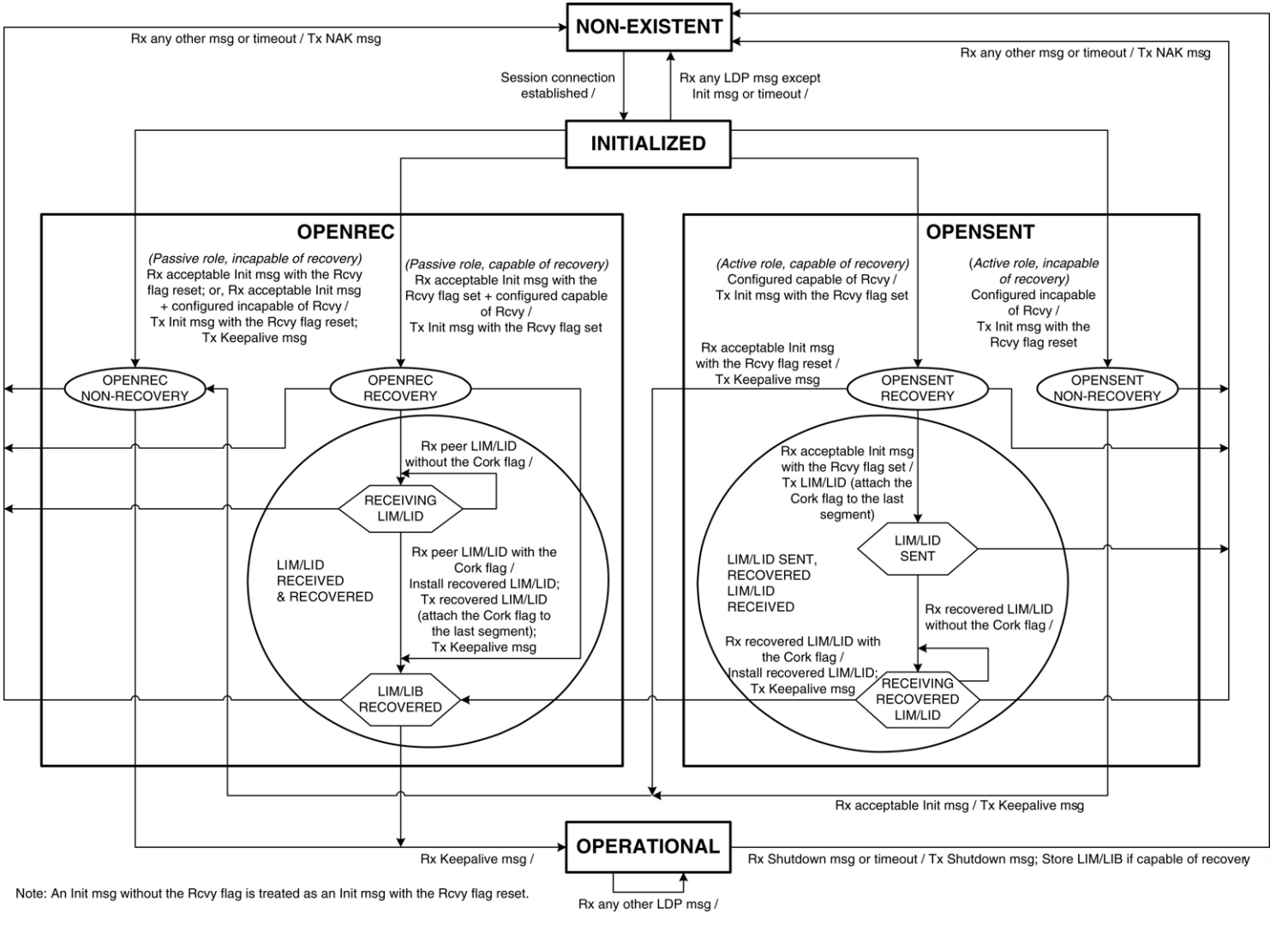

Fig. 4. Extended LDP initialization procedure for the fast LDP state information recovery.

The following is a description of the fast LDP state information recovery procedure. For simplicity, in the textual procedure listed below, we assume that an upstream node plays the active role in the initialization, i.e., an upstream node has a higher IP address for the LDP session than a downstream node. However, our proposed procedure is not limited to this. The same state machine specification can be applied when a downstream node plays the active role in the LDP initialization.

1. The upstream node attempts to restore its LIM for the outgoing link. If the label state information is preserved, it keeps the restored LIM. Otherwise, the upstream node sets the state of all labels to "presumably idle";
2. The upstream node advises the downstream node about the idle labels in its LIM by a series of LIM TLVs in an LDP session initialization message;
3. The downstream node attempts to restore its LID for the incoming link. If the label state information is preserved, the downstream node keeps the restored LID. Otherwise, the downstream node sets the state of all labels to "presumably idle";
4. After completely receiving the idle labels in the upstream node LIM, the downstream node calculates the idle labels agreed by the upstream and downstream nodes;
5. The downstream node updates its LID by changing the status of the labels calculated in the previous step to idle. If a label's state is "presumably idle", the state is changed to unknown;
6. The downstream node sends the idle labels back to the upstream node by a series of LID TLVs;
7. The upstream node updates its LIM by changing the status of the labels matching the received idle labels to idle. If a label's state is "presumably idle", the state is changed to unknown.

With the fast LDP state information recovery, a control node recovers its information about what labels are idle before a failure. A control node conservatively decides that a label is idle, since only when both upstream and downstream nodes agree that a label is idle, is the label considered as idle. If both nodes disagree on the state of a label, the label state is considered as unknown and will be further investigated in the next step LDP recovery – the detailed LDP state information recovery. The novelty of the fast LDP state information recovery is to quickly recover the basic label state (i.e., in-use or idle) first so that the LDP session can enter the operational state to handle new connection set-up requests without interfering with existing connections. The detailed label-related information and unknown state labels are left for the detailed LDP state information recovery. The recovery procedure has a merit that it is independent of the control plane failure type. It handles a control channel or a control node failure in a unified manner.

5.3. *A detailed LDP state information recovery*

We propose a detailed LDP state information recovery based on a query-and-reply approach, which extends the original LDP query procedure defined in [24]. The LDP Query message was used to gather a particular connection's information, e.g., labels used at each link along a connection, etc. We extend the LDP query by allowing a Query message to be propagated in either the upstream or downstream direction, allowing an intermediate control node to query both directions, and allowing a wide range of label-related information to be queried. With these extensions, an upstream node may recover the detailed LDP state information in its LIM by querying its downstream node. A downstream node may use the same procedure to recover the detailed LDP state information in its LID by querying its upstream node.

Our proposed detailed LDP state information recovery operates on a per label basis and in parallel to the normal LDP operation such as setting up or tearing down connections. The labels can be queried in any sequence. When a normal LDP operation requires a label's state information that has not been recovered or queried yet, a query about the label is sent immediately. In Fig. 5, a replaced control node is conducting the detailed LDP state information recovery. The replaced node has recovered the basic label state information after the fast LDP state information recovery. In Fig. 6, a connection teardown triggers an on-demand detailed LDP state information recovery related to a particular label. When certain detailed LDP state information is required in the replaced node, the node queries its peers to recover the information.

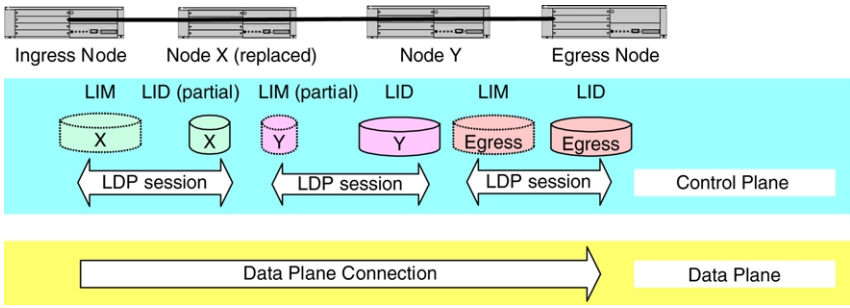

Fig. 5. A replaced node is conducting the detailed LDP state information recovery.

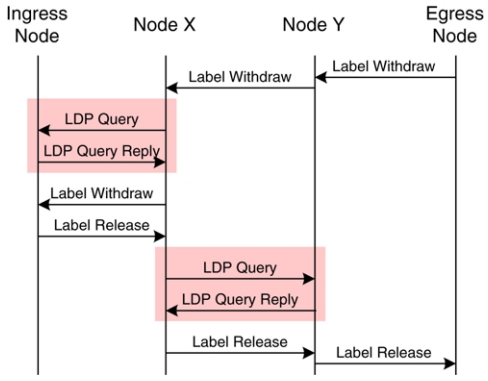

Fig. 6. A connection teardown triggers on-demand detailed LDP state information recovery related to particular labels.

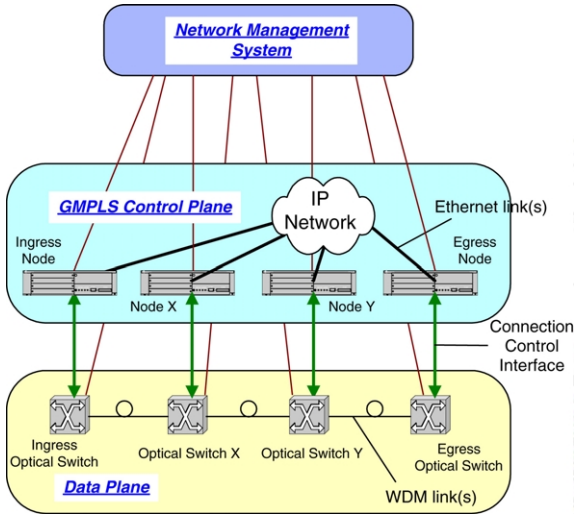

Fig. 7. A GMPLS-controlled WDM optical network.

6. Example

We present the proposed LDP failure recovery in a GMPLS-controlled WDM optical network (Fig. 7). The corresponding LDP sessions and redundant storage of the LDP state information are illustrated in Fig. 3. The recovery will be described with respect to the LDP session between the ingress node and node X after a failure and replacement of node X.

The LIM in the ingress node and the LID in node X are updated, as some wavelength channels between the ingress optical switch and optical switch X are allocated to lightpaths. As an example, Table 1 illustrates a snapshot of the LIM and LID contents at the moment before a failure of node X. Assume that node X fails and is replaced by a new node. The new control node gets its configuration first, then re-establishes an LDP session. The LDP configuration for the new control node must be the same as the failed control node. Any change in the LDP configuration will be detected by the network management system and result in disabling the automatic LDP recovery. During a control plane failure, no managed connection set-up or teardown is possible because the LDP signalling protocol is not functional. So the state of the data plane connections remains unchanged. An uncontrolled data plane connection failure during a control plane failure results in multiple failures, which cannot be recovered by our proposal. Before the new LDP session is successfully recovered and enters the operational state, if any other failure happens either in a data plane node or link under the control of this LDP session, or in a control plane element related to this LDP session, the network management system must turn off the LDP recovery. In this example, the failure of node X is the only failure. Without losing generality,

Table 1
Contents of the LIM in the ingress node and the LID in node X before a failure of node X

Port/fibre ID	Wavelength channel ID	Status	Operations at node X	Connection ID	……
A	1	Idle	None		
A	2	In-use	Cross-connect to output port P, Wavelength channel 2	Ingress node: Connection number 3	
B	1	Idle	None		
B	2	In-use	Cross-connect to output port Q, Wavelength channel 2	Ingress node: Connection number 5	

Table 2
Contents of the LID in node X after a replacement of the control node of node X

Port/fibre ID	Wavelength channel ID	Status	Operations at node X	Connection ID	……
A	1	Presumably idle	Unknown	Unknown	Unknown
A	2	Presumably idle	Unknown	Unknown	Unknown
B	1	Presumably idle	Unknown	Unknown	Unknown
B	2	Presumably idle	Unknown	Unknown	Unknown

Table 3
Contents of the LID in node X after the fast LDP state information recovery

Port/fibre ID	Wavelength channel ID	Status	Operations at node X	Connection ID	……
A	1	Idle	Unknown	Unknown	Unknown
A	2	Unknown	Unknown	Unknown	Unknown
B	1	Idle	Unknown	Unknown	Unknown
B	2	Unknown	Unknown	Unknown	Unknown

we assume the ingress node plays the active role in the LDP session initialization.

After a replacement of the control node of node X, the LID in node X is initialized. In this example, we assume no LDP state information before the failure is preserved. The proposed fast LDP state information recovery operates in the following steps:

1. The LID in node X is initialized, setting all labels to “presumably idle”. The contents of the LID are shown in Table 2;
2. The ingress node sends a list of idle labels in the ingress node’s LIM to node X. The list contains tuples (port/fibre ID, wavelength channel ID): (A, 1) and (B, 1);
3. After receiving the LIM TLVs, node X changes the status of labels (A, 1) and (B, 1) to idle, because these are the idle labels agreed by the ingress node and node X. The status of labels (A, 2) and (B, 2) are changed to unknown;
4. Node X sends a list of idle labels in node X’s LID to the ingress node. The list contains (A, 1) and (B, 1);
5. The ingress node updates the status of labels (A, 1) and (B, 1) to idle. There is no state change for labels (A, 2) and (B, 2).

Then the new LDP session enters its operational state. The LID in node X is recovered as shown in Table 3. New connection set-up requests can be processed. In the background, the LDP session continues the detailed LDP state information recovery. Node X gradually recovers its LDP state information that remains unknown by querying the ingress node. The ingress node replies to the queries by sending the backup LDP state information stored in the LIM to node X. While the detailed LDP state information recovery is ongoing, the new LDP session may process connection teardown requests. If a such request requires the LDP state information that has not been recovered yet, an on-demand query is to be conducted.

A similar LDP recovery is conducted for the LDP session between Node X and Y.

7. Performance analysis of the proposed LDP recovery

7.1. Notations

T_1^E	Average LDP encoding time for a label TLV;
T_1^D	Average LDP decoding time for a label TLV;
T_2^E	Average encoding time for the detailed LDP state information related to a label;
T_2^D	Average decoding time for the detailed LDP state information related to a label;
T^T	Average LDP message transmission time between two adjacent control nodes;
T_0^I	Average time for a standard LDP initialization without any LDP recovery;
T_1^I	LDP initialization time with a query-and-reply based LDP state information recovery;
T_2^I	LDP initialization time with the proposed two-step LDP state information recovery. All the idle labels are encoded into one single LIM/LID TLV;
T_3^I	LDP initialization time with the proposed two-step LDP state information recovery. A separate LIM/LID TLV is used to encode an idle label in the fast LDP state information recovery;
T^S	Additional connection set-up time;
T^R	Additional connection teardown time;
L	A control node's processing time for the detailed LDP state information related to a label;
α	A control node's processing load factor for the detailed LDP state information recovery;
μ	Average lifetime of an established connection;
H	Number of hops for a data plane connection;
n	Number of labels for an LDP session, i.e., the size of the label space for an LDP session;
m	Number of idle labels for a recovering LDP session;
R	Recovery time for the complete detailed LDP state information for all labels;
$P(t)$	Probability of an established connection that needs to be torn down at time t.

7.2. Comparison of the LDP initialization time

In addition to the standard LDP initialization time (T_0^I), the LDP recovery requires additional time in the LDP initialization. In this section, we will analyze the LDP initialization time. A query-and-reply based LDP state information recovery sets up a performance baseline. Encoding all labels into one or more LDP Query messages requires ($T_1^E \times n$) time. The Query message requires (T^T) time to be transmitted from a restarting control node to its peer control node where the backup LDP state information is stored. Then the peer control node requires ($T_1^D \times n$) time to decode the Query message, and ($T_1^E \times n + T_2^E \times n$) time to encode an LDP Query-Reply message. After (T^T) transmission time, the Query-Reply message arrives at the restarting control node, where it takes ($T_1^D \times n + T_2^D \times n$) time to be decoded. Note that the LDP encoding time includes the time searching the related databases, generating messages, etc. Similarly, the LDP decoding time includes the time parsing messages, populating the related databases, etc. Although a query-and-reply based recovery recovers the LDP state information after a standard LDP initialization, the restarting LDP session is not operational until all LDP state information is fully recovered, otherwise the existing connections may be disrupted by accidentally assigning in-use channels to new connections. To make a fair comparison, we consider the time to recover the LDP state information as part of the LDP initialization time. Thus, we obtain the LDP initialization time T_1^I,

$$T_1^I = T_0^I + (T_1^E + T_1^D) \times n \times 2 + (T_2^E + T_2^D) \times n + 2 \times T^T. \quad (1)$$

Our proposed fast LDP state information recovery only revives the information about what labels are idle. In the worst case, all labels are encoded in a LIM/LID TLV sending in one direction and only idle labels are encoded in a LIM/LID TLV sending in the other direction. Thus, the LDP initialization time is significantly reduced to T_2^I,

$$T_2^I = T_0^I + (T_1^E + T_1^D) \times (n + m) + 2 \times T^T. \quad (2)$$

Our proposed fast LDP state information recovery that is formally specified by a state machine (Fig. 4) further reduces the LDP initialization time by parallel processing the label encoding and decoding

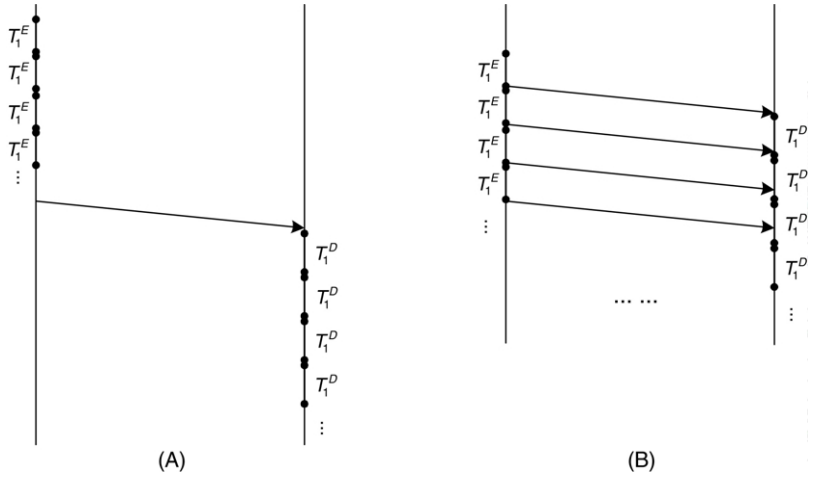

Fig. 8. A comparison of the serial and parallel processing of label encoding at an upstream node and decoding at a downstream node. (A) serial processing, and (B) parallel processing.

at an upstream node and a downstream node. A comparison between the serial processing and the parallel processing is shown in Fig. 8. Therefore, the LDP initialization time is further reduced to T_3^I,

$$T_3^I = T_0^I + \max(T_1^E, T_1^D) \times (n+m) + \min(T_1^E, T_1^D) \times 2 + 2 \times T^T. \quad (3)$$

Fig. 9 depicts a comparison of the LDP initialization time for different recovery mechanisms. T_1^E and T_1^D are 0.2 ms based on the experimental results on a prototype system for the GMPLS control plane. T_0^I is 2 s. T^T is approximately 1 ms for a metropolitan area application. No experimental results are available for T_2^E and T_2^D. We estimate T_2^E and T_2^D as 2 ms, which are around 10 times T_1^E and T_1^D. This estimation is for illustrative purposes. Different values for T_1^E and T_1^D result in the same trend as we show. The number of idle labels is assumed to be half of the total labels. The LDP initialization time is significantly reduced by our proposed LDP recovery.

7.3. Performance analysis of additional connection set-up and teardown time

Additional LDP state information is transferred from a LID to its backup LIM in the connection set-up phase. A downstream node encodes such information in an LDP Label Mapping message, which is decoded in an upstream node. Assuming all LDP sessions apply

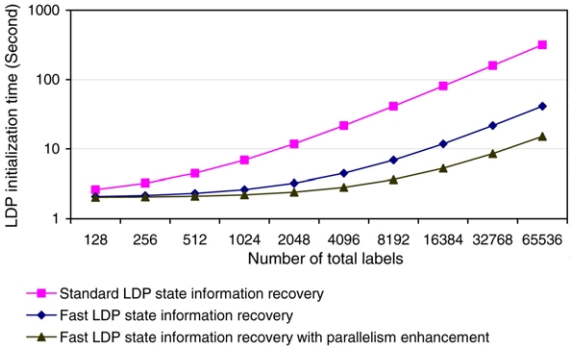

Fig. 9. A comparison of the LDP initialization time for different recovery mechanisms.

the backup mechanism, the total connection set-up time is increased by T^S time compared to no LDP state information backup. The additional connection set-up time T^S is

$$T^S = (T_2^E + T_2^D) \times H. \quad (4)$$

Using the previous estimations for T_2^E and T_2^D, for a three-hop connection, the additional connection set-up time is 12 ms.

After a fast LDP state information recovery, the restarting LDP session enters the operational state. While the restarting LDP session processes new LDP connection set-up requests, the detailed LDP state information is being recovered in the background. A control node's processing time L for the detailed LDP

state information related to a label is,

$$L = T_1^E + T_1^D + \max(T_2^E, T_2^D). \quad (5)$$

To minimize a performance degradation for the foreground LDP processing, the background process uses limited processing capacity. α is the processing load factor for the background LDP state information recovery process. We set $\alpha = 20\%$–40%. Therefore, the recovery time for the complete detailed LDP state information for all labels is,

$$R = \frac{n \times L}{\alpha}. \quad (6)$$

Connection teardown requests are processed while the detailed LDP state information is ongoing in the background. There is a possibility that a connection teardown request arrives before the detailed LDP state information is recovered. In such a case, an on-demand query-and-reply based LDP state information recovery is conducted. This induces additional connection teardown time T^R,

$$T^R = 2 \times [(T_1^E + T_1^D) \times 2 + (T_2^E + T_2^D) + 2 \times T^T]. \quad (7)$$

Using the previous estimations for T_2^E and T_2^D, the additional connection teardown time is 10 ms.

Assuming the lifetime of a data plane connection follows an exponential distribution, the probability of an established connection needing to be torn down at time t is,

$$p(t) = 1 - e^{-t/\mu}. \quad (8)$$

Fig. 10 shows the percentage of the connection teardowns that need to go through the extended connection teardown process shown in Fig. 6. The connection teardown requests are discarded before the restarting LDP session enters into the operational state. We use the previous estimations for T_2^E and T_2^D, and $\alpha = 24\%$. When the average lifetime of a data plane connection increases, and the number of labels decreases, it can be observed that the connection teardowns experiencing the additional connection teardown time T^R decrease.

7.4. Performance analysis of scalability, overhead and convergence

Our proposed LDP recovery is scalable thanks to the distributed backup of the LDP state information,

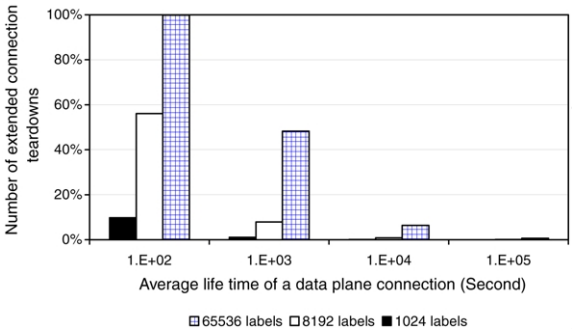

Fig. 10. Percentage of the connection teardowns that need to go through the extended connection teardown process.

and local storage and recovery between a pair of neighbour nodes in an LDP session. Our proposed LDP recovery can be selectively deployed on a per LDP session basis. The network management system or a human network operator may change the default configuration settings for the LDP initialization to enable or disable the LDP recovery. The backup storage and recovery of the LDP state information only involve a pair of nodes in an LDP session. When providing the LDP recovery for a critical control node, the LDP state information in the node is distributedly backed up in each of its upstream nodes. Each upstream node only takes care of the LDP state information related to the LDP session between the upstream node and the node being protected. Therefore, the performance and scalability bottleneck of a centralized network management system or backup information repository is removed.

The overhead associated with our proposed LDP recovery lies in two aspects: storage and processing overheads. In order to provide an LDP recovery, it is necessary to redundantly store all LDP state information. This storage overhead cannot be removed because our design goal is not to rely on the persistent storage of the LDP state information (or a message log) stored in a failed node itself. Our proposed LDP recovery synchronizes the LIM and LID in two neighbour nodes by piggybacking the additional LDP state information onto the regular LDP Label Mapping message. So there is no extra message exchanging phase in a connection set-up. The additional connection set-up time is modelled in the previous section.

The convergence of our proposed LDP recovery outperforms the standard query-and-reply based recovery. The fast LDP state information recovery recovers the minimal state information within one round of information exchanges, i.e., sending the (presumably) idle labels in one direction and then receiving the agreed idle labels in the other direction. The detailed LDP state information recovery takes significantly more time to accomplish. However, by conducting the detailed LDP state information recovery in the background, the negative performance impact is reduced.

8. Conclusions

With an LDP recovery, a restarting LDP session may recover the state information before a control plane failure. Such a capability enables the service continuation of the established data plane connections. With our proposed LDP failure recovery, any single-point failure can be fully recovered in a unified framework. Our approach does not need to diagnose the control plane failure type, and requires no additional hardware or modification to the data plane equipment.

Our proposed two-step LDP state information recovery achieves a good trade-off between processing load and speed. The fast LDP state information recovery only recovers the minimal LDP state information, which is what labels are idle before a control plane failure, within the shortest time. This feature allows a restarting LDP session to start processing connection set-up requests at its earliest time, without interfering with existing connections. The detailed LDP state information recovery runs in the background in parallel to the normal LDP operations. Only when a connection teardown requires the LDP state information that has not yet been recovered is an on-demand query based LDP state information recovery conducted.

Our proposed LDP recovery compliments a wide range of fault-tolerance and system failure recovery techniques. Different techniques need to be engineered to achieve the best performance. In particular, since our proposed LDP recovery is only able to recover from a single-point failure, we rely on the network management system to handle more complicated scenarios.

Acknowledgements

The authors would like to thank Dr. Delfin Montuno at Nortel Networks for his valuable discussions and comments on the original research. Abel Dasylva at Nortel Networks contributed to the performance improvement of the proposed mechanism by providing his experimental results based on a prototype system for the GMPLS control plane. Guo-Qiang Wang at Nortel Networks offered valuable comments. The authors appreciate the constructive comments from anonymous reviewers. Their suggestions are of great help to improve the quality of this research.

References

[1] G. Li, J. Yates, D. Wang, C. Kalmanek, Control plane design for reliable optical networks, IEEE Communications Magazine 40 (2) (2002) 90–96.
[2] M. Leelanivas, Y. Rekhter, R. Aggarwal, Graceful restart mechanism for label distribution protocol, IETF RFC 3478, February 2003.
[3] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella, Y. Rekhter, Generalized multiprotocol label switching: An overview of routing and management enhancements, IEEE Communications Magazine 39 (1) (2001) 144–150.
[4] J.P. Lang, J. Drake, Link management protocol (LMP), in: 16th National Fiber Optic Engineers Conference, NFOEC 2000, vol. 2, Denver, Colorado, August 2000, pp. 368–377.
[5] J. Lang (Ed.) Link management protocol (LMP), IETF draft draft-ietf-ccamp-lmp-10.txt, As of August 23, 2005, the status is RFC Editor Queue/State and will become a proposed standard.
[6] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, LDP specification, IETF RFC 3036, January 2001.
[7] B. Jamoussi (Ed.) Constraint-based LSP setup using LDP, IETF RFC 3212, January 2002.
[8] A. Farrel, Fault tolerance for the label distribution protocol (LDP), IETF RFC 3479, February 2003.
[9] H. Lam (Ed.) Distributed call and connection management: signalling mechanism using GMPLS CR-LDP, ITU-T recommendation G.7713.3/Y.1704.3, March 2003.
[10] J. Wu, D.Y. Montuno, H.T. Mouftah, G. Wang, A.C. Dasylva, Recovery from control plane failures in GMPLS-controlled optical networks, International Journal of Communication Systems 15 (7) (2002) 573–592.
[11] D. Griffith, A Comparison of RSVP-TE and CR-LDP, Optical Internetworking Forum (OIF) contribution OIF2000.179, August 2000.
[12] P. Brittain, A. Farrel, MPLS traffic engineering: A choice of signaling protocols, White paper of Data Connection Limited. Available: http://www.dataconnection.com (accessed June 2006).

[13] L. Fang, A. Atlas, F. Chiussi, K. Kompella, G. Swallow, LDP failure detection and recovery, IEEE Communications Magazine 42 (10) (2004) 117–123.
[14] A. Rodriguez-Moral, P. Bonenfant, S. Baroni, R. Wu, Optical data networking: Protocols, technologies, and architectures for next generation optical transport networks and optical internetworks, IEEE Journal of Lightwave Technology 18 (12) (2000) 1855–1870.
[15] J.W. Stewart III, BGP4: Inter-Domain Routing in the Internet, Addison-Wesley, Boston, 1999.
[16] S. Sanchez-Lopez et al., PNNI-based control plane for automatically switched optical networks, Journal of Lightwave Technology 21 (11) (2003) 2673–2682.
[17] D. Papadimitriou, J. Drake, J. Ash, A. Farrel, L. Ong, Requirements for generalized MPLS (GMPLS) signaling usage and extensions for automatically switched optical network (ASON), IETF draft draft-ietf-ccamp-gmpls-ason-reqts-07.txt, October 2004.
[18] E. Mannie (Ed.) Generalized Multi-protocol label switching architecture, IETF RFC 3945, October 2004.
[19] A.S. Tanenbaum, M. Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, New Jersey, 2002.
[20] A. Farrel, Applicability statement for restart mechanisms for the label distribution protocol (LDP), IETF RFC 2612, September 2003.
[21] J. Cucchiara, H. Sjostrand, J. Luciani, Definitions of managed objects for the multiprotocol label switching (MPLS), label distribution protocol (LDP), IETF RFC 3815, June 2004.
[22] C. Srinivasan, A. Viswanathan, T. Nadeau, Multiprotocol label switching (MPLS) label switching router (LSR) management information base (MIB), IETF RFC 3813, June 2004.
[23] T. Nadeau, and A. Farrel (Ed.) Generalized multiprotocol label switching (GMPLS) label switching router (LSR) management information base, IETF draft draft-ietf-ccamp-gmpls-lsr-mib-07.txt, February 2005.
[24] P. Ashwood-Smith, A. Paraschiv, D. Allan, Multi protocol label switching label distribution protocol query message description, IETF draft draft-ietf-mpls-lsp-query-09.txt, June 2003.