Building Reliable MPLS Networks Using a Path Protection Mechanism

Changcheng Huang, Carleton University
Vishal Sharma, Metanoia, Inc.
Ken Owens, Erlang Technology, Inc.
Srinivas Makam, Tellabs Operations, Inc.

ABSTRACT

It is expected that MPLS-based recovery could become a viable option for obtaining faster restoration than layer 3 rerouting. To deliver reliable service, however, MPLS requires a set of procedures to provide protection for the traffic carried on the label switched paths. In this article we propose a path protection mechanism that is simple, scalable, fast, and efficient. We describe in detail our design considerations, the communication of fault information to appropriate switching elements, and the fault detection protocol. In particular, we propose a reverse notification tree structure for efficient and fast distribution of fault notification messages.

INTRODUCTION

The migration of real-time and high-priority traffic to IP networks means that modern IP networks increasingly carry mission-critical business data, and must therefore provide reliable transmission. Current routing algorithms, despite being robust and survivable, can take a substantial amount of time to recover from a failure, which can be on the order of several seconds to minutes and can cause serious disruption of service in the interim. This is unacceptable for many applications that require highly reliable service, and has motivated network providers to give serious consideration to the issue of network survivability.

Path-oriented technologies, such as multiprotocol label switching (MPLS) [1], can be used to support survivability requirements to enhance the reliability of IP networks. In conventional IP forwarding, a router forwards an IP packet based on the longest match for the packet's destination IP address. As the packet traverses the network, each hop in turn forwards the packet by reexamining its destination IP address. In contrast to legacy IP networks, when a packet enters an MPLS network it is assigned a label. At subsequent hops the label is used as an index into a table that specifies the packet's next hop and a new label. The old label is swapped with the new label, and the packet is forwarded to its next hop. The path the packet traverses is therefore called a *label switched path* (LSP). Multiple LSPs can be merged at a specific node if packets from these LSPs are forwarded in the same manner (e.g., over the same downstream path, with the same forwarding treatment). This is called *label merging*.

In conventional forwarding, if it is desirable to force a packet to follow a particular route that is chosen explicitly rather than by the normal dynamic routing algorithm, the packet has to carry the encoding of its route (a list of IP addresses) along with it (source routing). This introduces significant overhead. In MPLS a label can be used to represent the route, so the identity of the explicit route need not be carried with the packet. This potentially allows MPLS networks to pre-establish protection LSPs for working LSPs and achieve better protection switching times than those in legacy IP networks.

A key feature of MPLS is that once the labels required for an LSP have been assigned through LSP setup or label distribution protocols, intermediate LSRs transited by the LSP do not need to examine the content of the data packets flowing on the LSP. Multiple labels can be placed on a packet to form a label stack that allows multiple LSPs to be tunneled, one within the other. The outermost LSP therefore becomes a tunnel that makes inner LSPs transparent to the intermediate LSRs, and therefore simplifies the forwarding tables at these LSRs. This feature is critical for the local repair approach described in the next section.

In this article we propose a path protection mechanism that is fast, scalable, efficient, and easy to implement. The rest of the article is organized as follows. We survey various recovery approaches. We highlight the main features of the proposed path protection mechanism. We examine each feature in detail, and conclude the article.

0163-6804/02/$17.00 © 2002 IEEE

Recovery Models

Recovery models can be classified according to the following two criteria: rerouting vs. protection switching, local repair vs. path protection.

Recovery by rerouting is defined as establishing new paths or path segments on demand for restoring traffic after the occurrence of a fault. The new paths may be based on fault information, network routing policies, predefined configurations, and network topology information. Thus, on detecting a fault, paths or path segments to bypass the fault are established using signaling. Reroute mechanisms are inherently slower than protection switching mechanisms, since more must be done following the detection of a fault. However, reroute mechanisms are simpler and more frugal since no resources are committed until after the fault occurs and the location of the fault is known [2]. Once the network routing algorithms have converged after a fault, it may be preferable, in some cases, to reoptimize the network by performing a reroute based on the current state of the network and network policies.

Protection switching recovery mechanisms pre-establish a recovery path or path segment, based on network routing policies, the restoration requirements of the traffic on the working path, and administrative considerations. The recovery path may or may not be link and node disjoint with the working path. However, if the recovery path shares sources of failure with the working path, the overall reliability of the construct is degraded. When a fault is detected, the protected traffic is switched over to the recovery path(s) and restored.

The two common protection switching recovery mechanisms are local repair (or link/node protection) and end-to-end[1] repair (or path protection). This section introduces these two mechanisms, and discusses their advantages and disadvantages.

Local Repair

The intent of local repair is to protect against a link or neighbor node fault, and to minimize the amount of time required for failure propagation. In local repair (also known as local recovery), the node immediately upstream of the fault is the one to initiate recovery (either rerouting or protection switching). Take, for example, Fig. 1. If link 1→2 fails, all the working paths that travel through link 1→2 and require protection can be switched to and stacked over a pre-established tunnel 1→6→2. While this bypass approach is simple for recovering from a link failure, it becomes much more complex to recover from a node failure. For example, if node 3 fails, two tunnels must be established to protect traffic streams that enter node 3 from node 2. Tunnel 2→7→5 can protect those working paths that travel through 2→3→5, while tunnel 2→8→4 can protect those working paths that travel through 2→3→4. If a node has N bidirectional links, there can be $N - 1$ different ways of forwarding for each ingress link. A total of $N^*(N - 1)$ different aggregate flows may exist within the node for the N ingress links. Because each flow requires a protection tunnel as shown in Fig. 1, it is easy to see that a minimum of $N^*(N - 1)$ tunnels must be set up to recover from its failure. If all the nodes within a network are equally likely to fail, this is clearly not a scalable solution. Therefore, although local repair is potentially the fastest to recover from a failure, it is only effective in situations where certain path components are much more unreliable than others, so only those components can be protected. Routing and bandwidth allocation algorithms for local repair can be found in [3] as an example.

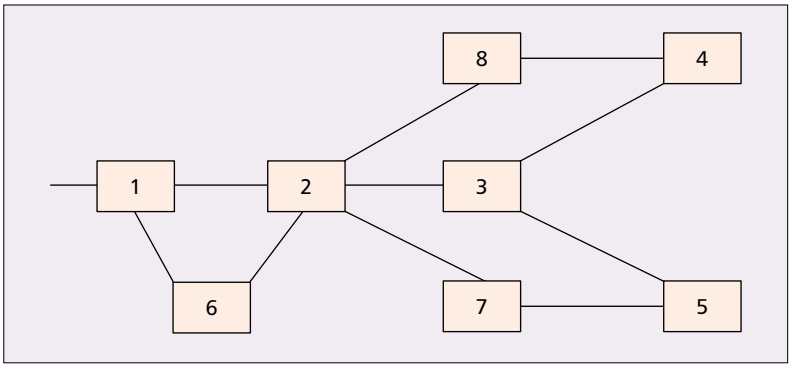

Figure 1. *An example of local repair.*

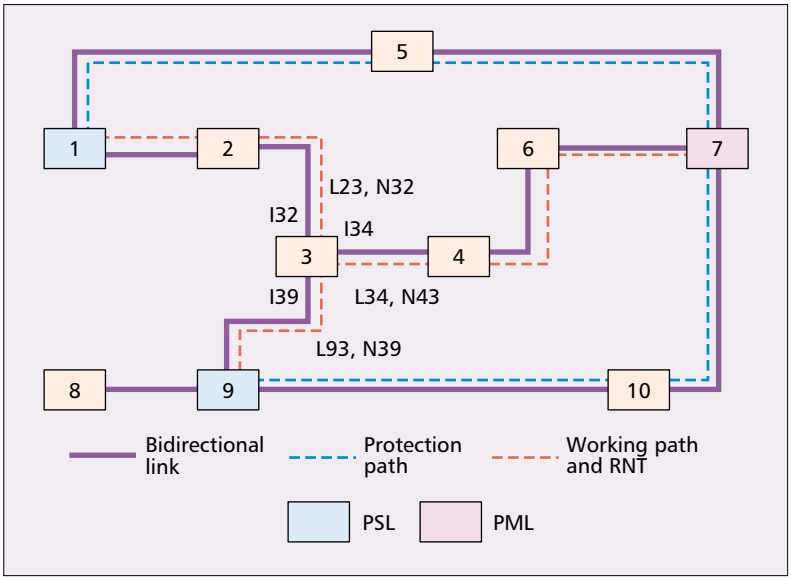

Figure 2. *An example of path protection.*

Path Protection

The intent of path protection is to protect against any link or node fault on a path or a segment of a path, which we call the *working path*. An LSR that is the transmitter of both the working path traffic and its corresponding recovery path traffic is called a *path switch* LSR (PSL). The PSL is the origin of the recovery traffic, but may or may not be the origin of the working traffic (i.e., the working traffic may be transiting the PSL). For example, path 1→2→3→4→6→7 in Fig. 2 is a path that requires end-to-end protection; therefore, node 1 will be the PSL. For path 8→9→3→4→6→7, only segment 9→3→4→6→7 requires protection, so node 9 is the PSL. In path protection, the recovery path can be made completely link and node disjoint with its corresponding working path.

This has the advantage of protecting against

[1] *In an MPLS domain, this may be more accurately characterized as edge-to-edge repair, because the portion of an LSP within the domain may be protected, even though the entire LSP may not be path protected.*

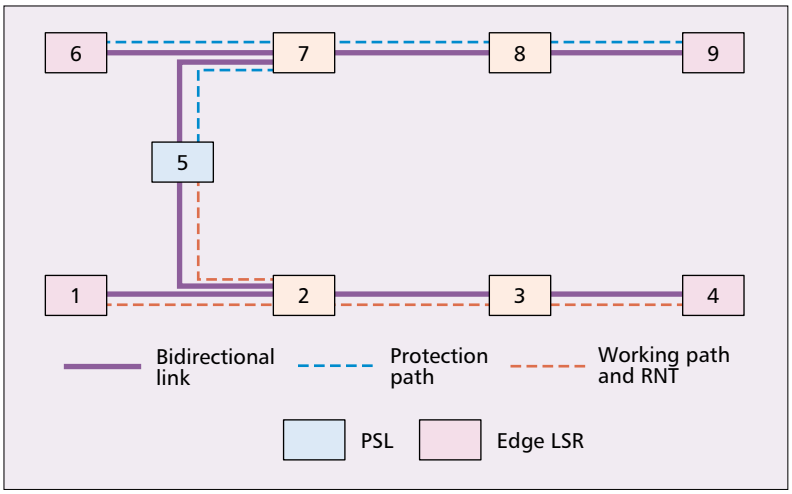

Figure 3. *An example of cross-domain protection.*

all link and node fault(s) on the working path. Assume an MPLS network has M edge nodes. The total number of recovery paths that have to be set up is proportional to $M^*(M-1)$. Therefore, if M is small, the path protection approach is more efficient than local repair. However, it is in some cases slower than local repair since it takes longer for the fault notification message to get to the protection switching point to trigger the recovery action. Simulation results have shown that the path protection approach can be significantly more efficient than local repair for wavelength-division multiplexing (WDM) [4] and asynchronous transfer mode (ATM) networks [5].

The resources (bandwidth, buffers, processing) on the recovery path may be used to carry either a copy of the working path traffic or extra traffic that is displaced when a protection switch occurs. This leads to two subtypes of path protection switching.

1+1 Path Protection

In 1+1 path protection, the resources on the recovery path are fully reserved, and carry the same traffic as the working path. Selection between the traffic on the working and recovery paths is made at the *path merge LSR* (PML) where the working and recovery paths converge. Therefore, it is PML-oriented. 1+1 protection can be very fast and simple because fault detection and protection switching happen at the same node. While it is easy to track the quality of the working path at the egress node (i.e., PML) in synchronous optical networks (SONET), this may not be the case for MPLS networks. Due to statistical multiplexing and traffic burstiness it may take a significantly long time for a PML to detect a fault. Furthermore, in some cases (e.g., multihoming), the working and protection path may not converge at all; therefore, a PML may not exist in certain MPLS networks (Fig. 3).

1:1 Path Protection

In 1:1 path protection (extendible to *m*:*n* protection), the resources allocated on the recovery path are fully available to preemptible low-priority traffic and can also be shared by other recovery paths, if their corresponding working paths are disjoint from the working path corresponding to the recovery path in question. In other words, in 1:1 protection the protected traffic normally travels only on the working path, and is switched to the recovery path when the working path has a fault. The protection switching is typically initiated by a PSL; therefore, this type of path protection is PSL-oriented. This method provides a way to make efficient use of the recovery path resources. The PML (if it exists) can simply merge the working and protection paths with MPLS label merging capability [1]. Under certain conditions different protection paths can share network resources. Reference [6] has shown that good coverage (in recovering from failures) can be achieved with minor degradation of network utilization. The routing and engineering of the primary and protection paths can be found, for example, in [7, 8].

Generally network operators aim to provide the fastest and best protection mechanism that can be provided at a reasonable cost. The higher the level of protection, the more resources are consumed. Therefore, it is expected that network operators will offer a spectrum of service levels. MPLS-based recovery should give the flexibility to select the recovery mechanism, choose the granularity at which traffic is protected, and also choose the specific types of traffic that are protected in order to give operators more control over that trade-off. With MPLS-based recovery, it can be possible to provide different levels of protection for different classes of service, based on their service requirements. For example, a virtual leased line (VLL) service or real-time applications like voice over IP (VoIP) may be supported using link/node protection together with pre-established prereserved path protection. Best effort traffic, on the other hand, may use established-on-demand path protection or simply rely on IP reroute or higher-layer recovery mechanisms. As another example of their range of application, MPLS-based recovery strategies may be used to protect traffic not originally flowing on LSPs, such as IP traffic, normally routed hop-by-hop, as well as traffic forwarded on LSPs.

In the following sections we discuss the proposed path protection approach designed specifically for MPLS networks. It can also easily be generalized to support MPLS-based optical networks. We will focus on the mechanisms that are essential for the operation of a path protection mechanism rather than routing and bandwidth allocation algorithms.

Key Features

A path protection mechanism typically consists of *protection configuration*, *fault detection*, *fault notification*, and *protection switching*. One of the major considerations in a path protection mechanism is to control the delay that must be incurred by the notification message traveling from the fault detection node to the protection switching node (i.e., PSL). This delay may cause packet loss and misordering. Our primary design goal, therefore, is to minimize this delay.

Some of the key features of our protection mechanism are:

A special tree structure to efficiently dis-

tribute fault and/or recovery information: Existing published proposals for MPLS recovery have not addressed the issue of fault notification in detail. Specifically, none of these proposals has discussed how to perform fault notification for the label merging case. In this article we propose a new fault notification structure called the *reverse notification tree* (RNT) that makes fault notification efficient and scalable.

A hello protocol to detect faults: Our assumption is that faults fall into different classes, and that different faults may be detected and corrected by different layers.

- Faults that are detected and corrected by lower-layer mechanisms (e.g., loss of signal or transmitter faults detected by SONET)
- Faults that are detected (but not corrected) by lower layers (e.g., failure of the reverse link) and may be communicated to the MPLS layer
- Faults that are not detected by lower layers (e.g., node failures or faults on the reverse link) and must be detected and corrected at the MPLS layer

Therefore, we adopt a hello protocol as a complementary fault detection mechanism.

A lightweight notification transport protocol to achieve scalability: Reliable transport mechanisms, such as TCP, are typically state-oriented and therefore difficult to scale. It is also very difficult to support point-to-multipoint communications based on reliable transport mechanisms. In our scheme, therefore, we propose a lightweight notification transport protocol to achieve scalability.

In this article we confine our discussion of protection to a single MPLS domain, and do not consider protection/recovery across multiple MPLS domains or multiple administrative boundaries. We note, however, that protection mechanisms in different domains may be concatenated, and (at least initially) these mechanisms may work autonomously, across the (possibly) multiple points of attachment between two adjacent domains. However, coordination of protection mechanisms across multiple domains or across multiple transport technologies is beyond the scope of this article. For example, in Fig. 3 working path 5→2→3→4 may extend beyond edge LSR 4, and its corresponding protection path 5→7→8→9 may also extend beyond edge LSR 9. While node 5 is the PSL, there is no PML in the domain. Similarly, working path 1→2→3→4 may extend beyond both edge LSRs 1 and 4, and its corresponding working path 6→7→8→9 may also extend beyond edge LSRs 6 and 9. There is no PSL or PML in the domain in this case.

In the following sections we will assume that the working and protection paths will not fail at the same time. This is because the probability that both paths will fail at the same time is very low. For those cases where this scenario is not negligible, it is not difficult to generalize our proposed mechanism by introducing extra protection path(s) to protect the first protection path where failures will be detected as the working path.

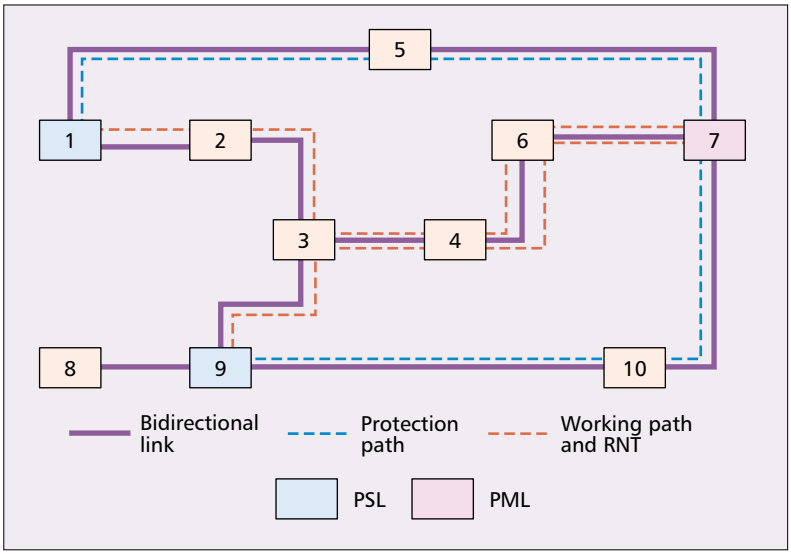

Figure 4. *An example of virtually merged LSPs.*

Reverse Notification Tree

Reverse Notification Tree Concept

Since LSPs are unidirectional entities and recovery requires the notification of faults to the LSR(s) responsible for switchover to the recovery path (the PSL), a mechanism must be provided for the propagation of fault and repair indications from the point of occurrence of the fault back to the PSL(s). These are called *fault indication signal* (FIS) and *fault repair signal* (FRS), respectively. The situation is complicated in the following two cases:

Physically merged LSPs: With label merging, multiple working paths may converge to form a multipoint-to-point tree, with the PSLs as the leaves. In this case, therefore, the fault indication and repair notification should be able to travel along a reverse path of the working path to all the PSLs affected by the fault. For example, in Fig. 2 for a fault along link 3→4 the affected PSLs are 1 and 9, whereas for a fault along link 2→3, the only affected PSL is 1.

Virtually merged LSPs: When several LSPs originating at different LSRs share a common segment beyond some node and a common identifier (e.g., the SESSION ID in RSVP-TE), we call such LSPs *virtually merged*. In this case also, savings in notification can be realized by sending a single notification toward the affected PSLs along segments shared by the LSPs emanating from these PSLs, and allowing the notification to branch out at the merge node(s). For example, in Fig. 4 for a failure along link 6→7 a single notification could be sent for working paths 1→2→3→4→6→7 and 8→9→3→4→6→7 along their common segment 7→6→4→3, although the two working paths may not be physically merged. The notification would branch out at node 3, which is the node where the LSP from node 1 to node 7 and the LSP from node 8 to node 7 virtually merge.

In both cases above, an appropriate notification path can be provided by the reverse notification tree (RNT), which is a point-to-multipoint

Our mechanism requires that the node upstream of the fault must be able to detect/learn about the fault. This motivates the need for a hello protocol, which allows a node upstream of the fault to detect the fault, either directly or implicitly.

Ingress label of RNT	Ingress interface of RNT	Egress label of RNT	Egress interface of RNT	Egress label of RNT	Egress interface of RNT
N43	I34	N32	I32	N39	I39

Table 1. *An example inverse label-swapping table for LSR 3 in Figure 2.*

tree that is an exact mirror image of the converged working paths, along which the FIS and the FRS travel (Fig. 2). There are several advantages to using an RNT:

- The RNT can be established in association with the working path(s) simply by making each LSR along a working path remember its upstream neighbor (or the collection of upstream neighbors whose working paths converge at the LSR and exit as one). Thus, no multicast routing is required.
- Only one RNT is required for all the working paths that merge (either physically or virtually) to form the multipoint-to-point forwarding path. The RNT is rooted at an appropriately chosen LSR along the common segment of the merged working LSPs and is terminated at the PSLs. All intermediate LSRs on the converged working paths share the same RNT. Therefore, the RNT enables a reduction in the signaling overhead associated with recovery. Unlike schemes that treat each LSP independently, and require signaling between a PSL and the PML individually for each LSP, the RNT allows for only one (or a small number of) signaling messages on the shared segments of the LSPs.
- The RNT can be implemented at layer 3, layer 2, or even layer 1 (e.g., through SONET K1/K2 bytes). In general, the lower the layer, the faster FIS will travel.

RNT Implementation

The RNT is used for propagating the FIS and FRS, and can be created by a simple extension to the LSP setup process. During the establishment of the working path, the signaling message carries with it the identity (address) of the upstream node that sent it (e.g., via the RECORD-ROUTE object in RSVP [9]). Each LSR along the path simply remembers the identity of its immediately prior upstream neighbor on each incoming link.

The node then creates an ''inverse'' crossconnect table for each protected outgoing LSP (or session, for virtually merged LSPs) and maintains a list of the incoming LSPs that merge into that outgoing LSP, together with the identity of the upstream node and the incoming interface through which each incoming LSP comes.

If the RNT is implemented by a point-to-multipoint LSP, the working path can be bound to the ingress label and interface of the RNT LSP at a LSR. By mapping the inverse crossconnect table to a label-swapping table, the RNT point-to-multipoint LSP can be set up using the information available in the inverse crossconnect table. The RNT ingress label and interface can then be used as an index into the label-swapping table to find the egress labels and interfaces of the RNT LSP, as shown in Table 1. Upon receiving an FIS, an LSR extracts the labels and checks its label-swapping table to determine the outgoing labels and interfaces, performs a label swap, and forwards the FIS to the appropriate upstream node(s). If the node is also a PSL, a copy of the FIS will be delivered to the upper layers of the node that are responsible for initiating the protection switching actions.

For example, consider Fig. 2, and assume that a layer 2 point-to-multipoint RNT, rooted at LSR 7 and extending to LSRs 1 and 9, is associated with the multipoint-to-point forward paths starting at LSRs 1 and 8 and terminating at LSR 7. Now in case of a fault on link 4→6, LSR 3 receives an FIS on the RNT in a labeled packet with label N43. It uses this label as an index into its label-swapping table, and learns that there are two previous nodes (namely those reachable via interfaces I32 and I39, respectively) to which the FIS needs to be forwarded. It encapsulates the received FIS into a labeled packet with labels N32 and N39, and dispatches them along interfaces I32 and I39, respectively.

If the RNT is implemented by a hop-by-hop layer 3 mechanism, using, for example, UDP packets (with a specific port number to identify the notification message type), the egress label and interface of the working path can be used as an index into the inverse crossconnect table to obtain the IP addresses of the previous hop(s) and the associated outgoing interface(s), as illustrated in Table 1. On each hop, the FIS carried in the UDP packet carries the label and interface of the working path for that hop. Thus, if the receiving node is not a PSL, the label and interface in the FIS can be extracted and used to access the inverse crossconnect table. The label and interface used by the working LSP on the hop(s) to the upstream node(s) are then inserted into FIS packet(s), and the FIS packet(s) transmitted to the appropriate upstream node(s) along the interface specified in the inverse crossconnect table.

As in the example above, in case of a fault on link 4→6, LSR 3 receives an FIS from LSR 4 that contains the outgoing label L34 (carried in the payload of FIS) and the outgoing interface I34 of the LSP affected by the fault. LSR3 uses these to index its inverse crossconnect table (Table 2), and learns, as before, that there are two previous nodes (those reachable via interfaces I32 and I39, respectively) that must receive an FIS. It then creates two FIS packets as follows. The first, for transmission along interface I32, contains the label L23 used by LSR 2 to transmit data to LSR 3 along the working LSP. The second, for transmission along interface I39, contains the label L93 used by LSR 9 to transmit data to LSR 3 along the working LSP. The two FIS packets are then sent as IP packets to their corresponding next hops.

Egress label of working path	Egress interface of working path	Next-hop IP address of RNT	Egress interface of RNT	Ingress label of working path
L34	I34	I9	I39	L93
		I2	I32	L23

Table 2. *An example inverse crossconnect table for LSR 3 in Figure 2 using a hop-by-hop (layer 3) RNT.*

Hello Protocol for Fault Detection

Each LSR must be able to detect certain types of faults, such as path failure (PF), path degraded (PD), link failure (LF), and link degraded (LD), and propagate an FIS message toward the PSL. Here we consider unidirectional link faults, bidirectional (or complete) link faults, and node faults.

Our mechanism requires that the node upstream of the fault must be able to detect/learn about the fault. This motivates the need for a hello protocol, which allows a node upstream of the fault to detect the fault, either directly or implicitly. Our hello protocol is similar to those used in Open Shortest Path First (OSPF) [10]. The reason we need a separate hello protocol is that the timers in routing protocols are typically set to relatively large values compared to what is needed for a recovery mechanism. Also, the fault detection mechanism must provide the trigger for generating the FIS. Our hello protocol provides a complementary mechanism to all existing fault detection mechanisms such as physical layer fault detections through liveness messages exchanged between neighboring LSRs. Each LSR sends liveness messages periodically to its neighbors. A liveness message will carry the ID of the LSR and all the IDs of its neighbors discovered through the liveness messages sent by its neighbors. An LSR can learn a bidirectional link is working properly if it sees its own ID in the liveness message sent by the LSR on the other end of the link.

Lightweight Transport Protocol

The notification is based on the transmission of packets that are sent periodically until the nodes responsible for switchover learn of the fault. Since no acknowledgments or handshaking between adjacent nodes are needed, the mechanism works only with timers and does not require the maintenance of state.

The rapid notification of a fault is effected by the propagation of the FIS message along the RNT. Due to the timers built into the FIS/FRS propagation mechanism, the transportation of FIS/FRS messages does not require a reliable mechanism like TCP.

Any LSR may generate an FIS. The node that initiates the FIS will continue to send FIS messages at an interval of *t1* (set by a timer) until another timer *t2* expires. After *t2* expires it is assumed that either upper-layer protection will have been triggered (therefore, MPLS layer protection is not necessary anymore) or a large enough number of FIS messages will have been sent to reach the desired reliability in conveying fault information to the PSL(s).

The purpose of timer *t1* is to control the tradeoff between notification delay of the FIS and the resources consumed when sending the FIS. If *t1* is too large, it may take a relatively long time for the node that initiated the FIS transmission to send the second FIS if the first FIS message is lost, thereby increasing notification delay. On the other hand, if *t1* is small, the repetitive sending of FIS messages may waste bandwidth and processing power because the first message may already have reached the PSL(s).

Conclusion

In this article we proposed a PSL-oriented path protection mechanism that consists of three components: a reverse notification tree, a hello protocol, and a lightweight notification transport protocol. One of the key issues in any path protection mechanism is the delay experienced by an FIS message. Our mechanism minimizes this delay by:

- Building a fast and efficient notification tree structure
- Using a lightweight transportation mechanism for that notification message, which reduces the overhead associated with handshaking and other synchronization requirements

The RNT can be implemented in layer 2 or even layer 1 to further reduce the delay. The implementation of the RNT is simple and straightforward, since it does not require any extra routing and requires only very simple calculations.

The notification is based on the transmission of packets that are sent periodically until the nodes responsible for switchover learn of the fault. Since no acknowledgments or handshaking between adjacent nodes is needed, the mechanism works only with timers and does not require the maintenance of state.

Acknowledgments

We would like to thank members of the MPLS WG list, in particular Dave Allan, Bora Akyol, Neil Harrisson, Ping Pan, and J. Noel Chiappa, for suggestions and feedback.

References

[1] Rosen, E., Viswanathan, A., and Callon, R., "Multiprotocol Label Switching Architecture," IETF RFC 3031, 2001.
[2] G. Ahn and W. Chun, "MPLS Restoration Using Least-Cost Based Dynamic Backup Path," P. Lorenz (Ed.): ICN 2001, LNCS 2094, Springer, 2001, pp. 319–28.
[3] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregate Link Information," *Proc. IEEE INFOCOM '01*, pp. 376–85.
[4] B. N. Van Caenegem, N. Wauters, and P. Demeester, " Spare Capacity Assignment for Different Restoration Strategies in Mesh Survivable Networks," *Proc. ICC '97*, pp. 288–92.
[5] T. Frisanco, "Optimal Spare Capacity Design for Various Switching Methods in ATM Networks," *Proc. ICC '97*, pp. 293–98.
[6] S. Han, and K. G. Shin, "Fast Restoration of Real-Time Communication Service from Component Failures in Multihop Networks," *Proc. ACM SIGCOMM '97*.
[7] M. Kodialam, and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," *Proc. IEEE INFOCOM '00*, pp. 902–11.
[8] J. Veeraswamy, S. Venkatesan, and J. C. Shah, "Spare

We propose a PSL-oriented path protection mechanism that consists of three components: a reverse notification tree, a hello protocol and a lightweight notification transport protocol.

Capacity Assignment in Telecom Networks Using Path Protection," *Proc. 3rd Int'l. Wksp. Modeling, Analysis, and Simulation (MASCOTS)*, 1995.
[9] D. Awduche *et al.*, "Extensions to RSVP for LSP Tunnels," Internet Draft, work in progress, draft-ietf-mpls-rsvp-lsp-tunnel-07.txt, Aug. 2000.
[10] Moy, J., "OSPF Version 2," IETF RFC 2328, Apr. 1998.

ADDITIONAL READING

[1] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregate Link Information," *Proc. IEEE INFOCOM '01,* pp. 376–85.

BIOGRAPHIES

CHANGCHENG HUANG (huang@sce.carlton.ca) received his B.Eng. in 1985 and M.Eng. in 1988, both in electronic engineering, from Tsinghua University, Beijing, China. He received a Ph.D. degree in electrical engineering from Carleton University, Ottawa, Canada in 1997. He worked for Nortel Networks, Ottawa, Canada from 1996 to 1998 where he was a systems engineering specialist. From 1998 to 2000 he was a systems engineer and network architect in the Optical Networking Group of Tellabs, Illinois. Since July 2000 he has been an assistant professor in the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada. His research interests include self-similar traffic modeling, congestion control algorithms, Internet architecture, routing, and control algorithms for optical networks.

VISHAL SHARMA is principal consultant at Metanoia Inc., specializing in technical consulting for telecom vendors and service providers. He focuses on technology strategies, system and network architectures, detailed software/hardware architecture and design trade-offs, competitive analysis, product development, and knowledge enhancement for Metanoia's clients. He has seven patents filed in the areas of MPLS recovery, high-speed switch architectures, switch/router scheduling, and optical routing. He earned a B.Tech. degree from the Indian Institute of Technology, Kanpur, and M.S. and Ph.D. degrees from the University of California, Santa Barbara. He has over ten years of diverse research and industry experience, which includes work at UC Santa Barbara, Motorola, the Multidisciplinary Optical Switching Technology Center at UCSB, Tellabs, ACT Networks, Digital Instruments, Jasmine Networks, Mahi Networks, and Cariden Technologies, Inc. He is active in the IETF and OIF, and is an active speaker and participant in several industry fora.

Srinivas Makam has over 18 years of research and industry experience in telecommunications which includes work at Tellabs, DSC Communications, AT&T Bell Labs, and UCLA. His research interests include optical networking, multi-service switching platforms, wireless access transport networks for 3G, and GMPLS. He earned a B.S. (E.E.) degree from Bangalore University, India, an M.S. (C.S.) degree from Sangamon State University, Illinois, and a Ph.D. (C.S.) degree from the University of California in Los Angeles. He has participated in IETF, ATM Forum, and T1X1 meetings as well as being an active speaker at various industry conferences.

KENNETH ROBERT OWENS JR. is currently a principal design engineer at Erlang Technology, Inc. He has made significant contributions in the network processor FPGA design and simulation area. Prior to Erlang, he spent five years as a senior data network engineer in the Optical Networking Group of Tellabs. He made significant contributions to the Network System Engineering group by defining next-generation architectures of ATM and IP networks and equipment, ATM and IP functional modeling, and the validation of layered systems. Prior to Tellabs, he spent one and a half years at MCI Worldcom Advanced Technology Group researching, developing, and evaluating ATM and IP devices. He earned an M.S. degree in electrical engineering from the University of Missouri-Rolla in December 2001.