Efficient Distributed Restoration Path Selection for Shared Mesh Restoration

Guangzhi Li, *Member, IEEE*, Dongmei Wang, *Member, IEEE*, Charles Kalmanek, *Member, IEEE*, and Robert Doverspike, *Senior Member, IEEE*

***Abstract*—In MPLS/GMPLS networks, a range of restoration schemes will be required to support different tradeoffs between service interruption time and network resource utilization. In light of these tradeoffs, path-based end-to-end shared mesh restoration provides a very attractive solution. However, efficient use of bandwidth for shared mesh restoration strongly relies on the procedure for selecting restoration paths. In this paper, we propose an efficient restoration path selection algorithm for restorable connections over shared bandwidth in a fully distributed MPLS/GMPLS architecture. We also describe how to extend MPLS/GMPLS signaling protocols to collect the necessary information efficiently. To evaluate the algorithm's performance, we compare it via simulation with two other well-known algorithms on a typical intercity backbone network. The key figure of merit for restoration bandwidth efficiency is *restoration overbuild*, i.e., the extra bandwidth required to meet the network restoration objective as a percentage of the bandwidth of the network with no restoration. Our simulation results show that our algorithm uses significantly less restoration overbuild (63%–68%) compared with the other two algorithms (83%–90%).**

***Index Terms*—GMPLS, MPLS, optical network, RSVP-TE, shared mesh restoration.**

I. Introduction

FAST restoration of service after a network failure is a crucial aspect of current and future IP and transport networks. Recently, there has been a great deal of work addressing restoration functionality in both Multi-Protocol Label Switched (MPLS) networks [5], [6] and transport networks that use Generalized MPLS (GMPLS) protocols [7]–[9]. The GMPLS architecture addresses signaling protocols for establishing connections, known as label switched paths (LSPs) in both packet and transport networks. Due to the rapid development of GMPLS, shared mesh restoration is attracting a lot of attention from both academia and industry [1], [12], [15], [17], [22]–[25], [27]. A key issue in the design of a shared mesh restoration scheme is the restoration path selection algorithm. There is a large body of literature on centralized algorithms to select restoration paths [15], [16], [18], [19], [27]. However, there has been very little work to date addressing how to select the restoration path for restorable LSPs in a distributed manner [1], [22]. This paper proposes an efficient distributed shared restoration path selection algorithm for path-based end-to-end restorable LSPs, and addresses signaling protocol extensions and operational procedures to realize this end-to-end path-based shared mesh restoration approach. We compare the bandwidth utilization and message overhead of our approach with two other well-known distributed restoration path selection algorithms. Our work focuses primarily on transport networks with "channelized" link bandwidth where TDM or optical connections (such as SONET OC-48, wavelengths, etc.) are established over cross connects, especially optical cross connects (OXCs). Since the term ***connection*** applies to a broader class of networks, this paper will use ***connection*** instead of LSP. Note that while our simulation results apply primarily to such channelized networks, the approach is also applicable to shared mesh restoration with guaranteed bandwidth LSPs in packet-based MPLS networks.

There are many approaches to network recovery, supporting a range of tradeoffs between network resource utilization (cost) and service interruption time [14], [20], [21], [24]. Clearly, it is important to minimize service interruption time, but schemes achieving this usually do so at the expense of network resource utilization. The result is increased cost to the service providers. Different restoration schemes operate with different tradeoffs between spare bandwidth requirements and service interruption time.

In light of these tradeoffs, service providers are expected to support a range of different services. One important service differentiator is the service interruption time in the event of network failures. For example, a service provider's highest offered service level would generally ensure the most rapid recovery from network failures. However, services with the fastest recovery schemes (e.g., $1 + 1$, or 1:1 protection) typically use a large amount of spare restoration bandwidth, and are thus not cost effective for most customer applications. Significant reductions in spare bandwidth can be achieved by instead sharing this bandwidth across multiple restoration paths.

To minimize restoration delay after a failure, we assume that the restoration path is preselected and is physically diverse (i.e., link/node disjoint) from the service path [3]. However, restoration resources (channels) are prereserved but not selected until after failure occurs. Thus, resources can be *shared* by multiple restoration paths whose service paths are not susceptible to simultaneous failure. Since we address distributed restoration path selection, link usage information needs to be distributed throughout the network. Some of the required information is provided by link state routing advertisements current provided in GMPLS. In this paper, we propose signaling protocol extensions to distribute additional

Manuscript received April 16, 2002; revised September 9, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Orda.

The authors are with AT&T Research Labs, Florham Park, NJ 07932 USA (e-mail: gli@research.att.com).

Digital Object Identifier 10.1109/TNET.2003.819071

link usage information among network nodes. Our proposal keeps the overhead of distributing the additional link state information within acceptable bounds. This paper considers the case of restoration against single failures only, as is common in practice. The single failure constraint is reasonable because a restoration path is likely to be used only for a short time until the failure is repaired or a new service path is established. Furthermore, most transport service providers do not plan for restoration from multiple, simultaneous link or node failures because it would be prohibitively expensive to protect against events that generally have a very low probability of occurrence.

To demonstrate the efficiency of our algorithm, we evaluate the bandwidth utilization and message overhead of two other well-known distributed restoration path selection algorithms, which we refer to as Shortest Path Restoration (SPR) [18] and Partial Information Restoration (PIR) [1], [12]. Since our algorithm has full knowledge of bandwidth that is shared among restoration paths, we call it Full Information Restoration (FIR). Each of these algorithms allows some amount of bandwidth sharing among restoration paths, however, the algorithms require different amounts of network state information to be distributed. Thus, we explore the potential bandwidth savings that can be achieved by having more information distributed to network nodes. Our simulation is based on a realistic network topology. The results demonstrate that there is only marginal difference between SPR and PIR in terms of their bandwidth utilization, while FIR reduces the amount of reserved restoration bandwidth dramatically. This represents a potentially significant cost savings to network providers. The additional information needed for the FIR algorithm does not require extra link state information to be flooded via the routing protocol, such as Open Shortest Path First (OSPF). The additional information can be efficiently distributed via signaling extensions. Thus, we believe that FIR is suitable for use in real transport networks.

The paper is organized as follows. Section II presents the problem definition and introduces the restoration path selection algorithms. Section III gives an operational overview of the FIR signaling mechanism as defined to handle single link failures. Section IV extends the signaling to handle node and fiber span failures. Sections V and VI present the simulation scenarios and performance results, respectively. In Section VII, we provide two potential alternatives to the distributed restoration path selection scheme to further reduce the network overbuild.

II. Overview

A. Design Objectives

Assume a GMPLS backbone network represented by the graph $G = (V, E)$, where V is the set of network nodes (e.g., switches or routers) and E is the set of network links. Connection requests originating at a client arrive at the source node and connections are established or deleted in a distributed manner, via signaling among the nodes. For each restorable connection request, the source node needs to compute a service path Ps and a restoration path Pr in the network. In general, multiple connections will share a network link, hence, a single link failure can cause multiple connections to fail simultaneously. The design objectives for Ps and Pr are as follows.

1) Ps and Pr should be link/node disjoint.
2) The required bandwidth associated with different restoration paths should be shared if their service paths are not subject to simultaneous failures. For example, assume b bandwidth (channels) can be used on link k for restoration path $Pr1$ (protects service path $Ps1$). b can be shared by restoration path $Pr2$ that protects service path $Ps2$ as long as $Ps1$ and $Ps2$ cannot fail simultaneously.
3) Enough bandwidth must be reserved on all links in the network such that for any single failure, there is enough bandwidth to restore every affected service path.
4) The total bandwidth reserved for restoration over the whole network should be minimized.

Within this distributed context, it is not feasible to use centralized algorithms designed for capacity planning to compute optimized service and restoration paths. Rather, since connection requests arrive one at a time at a source node, the source node is required to make the routing decision without knowledge of future requests. If the network does not have enough bandwidth available for either the service path or the restoration path, the request should be rejected without readjusting existing connections. The restoration path should be selected to be physically diverse from the service path. We first focus on single link failures, then discuss how to extend the approach to single node failures and single fiber span failures, which can affect multiple links.

B. Shared Reservations

A restorable connection (LSP) in a GMPLS network supporting shared mesh restoration has both a service path and a restoration path. During normal network operation, the connection is established along the service path, with resources reserved along the restoration path. To realize shared restoration, bandwidth (e.g., a set of channels) is reserved along the restoration path during service path provisioning. The bandwidth reserved on each link along a restoration path is not dedicated to particular connections, but rather shared across multiple restoration paths whose service paths are not expected to fail simultaneously. Furthermore, this restoration bandwidth may be used to support pre-emptable (hence, usually less expensive) connections that can be disrupted if the bandwidth is needed for restoration. The bandwidth reserved on the restoration paths must be sufficient to recover all affected restorable connections in the event of any single failure. This requires the specific failure events to be identified when selecting and reserving bandwidth along the restoration path. To allow bandwidth to be shared among link-disjoint service paths and when we are protecting against link failures, nodes along the restoration path need to know the links along the service path when reserving bandwidth. Similarly, to allow bandwidth to be shared among node-disjoint service paths and if we are protecting against node failures, the nodes along the restoration path need to know the nodes along the service path. In both situations, bandwidth can be shared on common links of multiple restoration paths as long as their service paths are not subject to simultaneous failures.

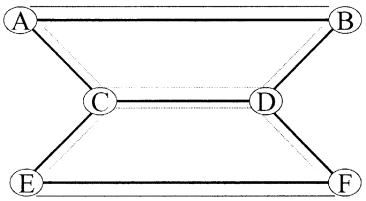


Fig. 1. Shared reservation.

Fig. 1 shows a simple example illustrating a shared reservation. The figure shows a network with six nodes and seven links. Suppose a connection request asking for one unit of bandwidth from A to B arrives at node A. Node A selects A-B for the service path and A-C-D-B for the restoration path. When this connection is established, one unit of bandwidth is allocated on link AB, and one unit of bandwidth is reserved on each of the links AC, CD, and DB. Subsequently, another connection request asking for one unit of bandwidth from E to F arrives at node E. Node E selects E-F for the service path and E-C-D-F for the restoration path. In this example, when reserving resources along the restoration path, it is unnecessary to reserve an additional unit of bandwidth on link CD because the two service paths AB and EF are link/node disjoint, i.e., they are not subject to simultaneous failure under the single failure assumption. In addition, the reserved channels are not dedicated to particular connections, but rather are allocated only after a failure has occurred. Thus, a total of five units of bandwidth are reserved to protect both service paths against any single link failure whereas without shared reservations, a total of six units of bandwidth would be needed. Sharing the reserved bandwidth among the restoration paths reduces the total reserved bandwidth required.

C. Routing Information Exchange

In GMPLS networks, a link state routing protocol, such as OSPF or Intermediate System-Intermediate System protocol (IS-IS), is used to distribute network topology information to each node in a particular network routing domain. Traditionally, these protocols have only propagated link up/down status among the nodes. To support path computation in GMPLS networks, both OSPF and IS-IS have been extended to propagate additional information about each link, such as available bandwidth, bandwidth in service, etc. [10], [11]. This information is used to select the service path for each connection request. However, to support restoration path selection, some algorithms require extra information to be carried by routing and signaling protocols.

Most path selection algorithms in current networks are based on Dijkstra's algorithm, which selects a path of minimum weight among all suitable paths. The assignment of link weights provides a "knob" to control path selection. In this section, we describe link state information that is distributed in order to compute the link weights. The key idea is that the weights used to compute the restoration path should take into account shared restoration bandwidth. In the next section, we describe three path selection algorithms in detail that rely on the information described here.

We assume that the following link state information is flooded by the routing protocol. Note that information items 1, 3, and 4 are available in current extensions of OSPF/IS-IS for GMPLS. It also should be possible to distribute the information in item 2 using existing OSPF/IS-IS traffic engineering extensions.

1) Service bandwidth $(S[i])$: This is the total amount of bandwidth used by service LSPs on each link.
2) Reserved bandwidth $(R[i])$: This bandwidth is not used when the network is in a non-failure condition, but may be used by the restoration process upon failure.
3) Available bandwidth $(A[i])$: This bandwidth is free to be allocated to service bandwidth or reserved bandwidth as new connection requests arrive on each link. Note that the total bandwidth on link i is $S[i]+R[i]+A[i]$. We assume that these three types of bandwidth share a common bandwidth pool.
4) Administrative Weight $(W[i])$: This quantity is set by the network operator and may be used as a measure of the heuristic "cost" to the network. A path with a smaller total weight is typically preferable. One popular administrative weight is a constant = 1, i.e., path weight equals hop count. A path with smaller hop count consumes fewer network resources than a larger one. Another possible administrative weight may reflect the link mileage. The actual cost of a transport connection may be considered to be roughly related to a fixed cost plus a cost that is proportional to the length in miles of the path.

D. Restoration Path Selection Algorithms

We assume that the service path is selected as the shortest path based on the administrative weights. Before addressing restoration path selection, we describe procedures for sharing reservation bandwidth that are common to all of the restoration path selection algorithms that we study. For each link, there is a *master node* which is responsible for maintaining the link status. This node could be the node terminating the link having the smaller node id. Assume that the master node for link k maintains a local array $Failother(k)[i]$, $i \in E$, where $Failother(k)[i]$ is the amount of bandwidth required on link k to restore all failed connections if link i fails. (Section III describes in detail how to maintain the $Failother(k)$ array for link k along the restoration path dynamically.) Briefly, when the restoration path is selected, a reservation message carrying a list of the links on the service path Ps is sent along the restoration path. This allows the master node of link k along the restoration path to automatically update the corresponding $Failother(k)[i]$ entry. Since the reservation message contains the links of Ps, each node along the restoration path has the information allowing it to share reservations. Note that this scheme can also be extended to shared risk groups (SRGs), such as fiber spans, over which multiple links can be routed [14].

Now, we are ready to address three algorithms that may be used by the source node for restoration path selection. The critical issue addressed in this paper is how to collect the necessary information to select the restoration path in order to minimize the total reserved restoration resources over all network links. We call the simplest solution *Shortest Path Restoration* (SPR). SPR selects a restoration path using only the administrative weight and the reserved bandwidth information $R[k] =$

$max\{Failother(k)[i] : i \in E\}$ for each link, which are distributed by the extended routing protocol. By reserving a bandwidth equal to $R[k]$ on each link k for restoration, SPR ensures that there are sufficient resources reserved to protect against any single link failure. After the service path is selected, the source node excludes the links of the service path, as well as any link with insufficient available bandwidth, from the network topology G. The source node then uses a shortest-path-first algorithm to select the restoration path. SPR is simple and uses the minimum link state information which is provided by current OSPF/IS-IS extensions. However, SPR does not consider optimal bandwidth sharing in its restoration path selection algorithm and, therefore, the selected path may not reflect the maximum possible bandwidth sharing.

Kodaliam *et al.* present another solution [1], [12], which we refer to as *Partial Information Restoration* (PIR). The basic idea behind PIR is to weight each link using an estimate of the additional bandwidth that needs to be reserved if a particular restoration path is selected. After the service path Ps is selected, the source node computes the maximum service bandwidth M over all links along the service path, i.e., $M = max\{S[i], i \in Ps\}$. Then the source node assigns a weight to each link in the network:

$$w[i]=\begin{cases}\min(b, M+b-R[i]) \\ \qquad \cdot W[i], & \text{if } M+b-R[i]>0 \text{ and } i\notin Ps \\ \varepsilon, & \text{if } M+b-R[i]\leq 0 \text{ and } i\notin Ps \\ \infty, & \text{if } i \in Ps\end{cases}$$

where b is the size of the bandwidth request (e.g., number of channels) and ε is a small number (ε is used instead of 0 so that among multiple paths with 0 weight, the minimum hop path will be chosen).

The source node then uses Dijkstra's algorithm to select the restoration path Pr that minimizes $\Sigma w[i]$ for i in Pr. The idea of the term $\min(b, M + b - R[i])$ is to capture the amount of additional restoration bandwidth needed if, upon failure of the service path Ps, the connection is routed over a restoration path that contains link i. This scheme requires very little additional information compared with SPR and, in general, chooses the links with the largest reserved restoration bandwidth to avoid increasing $R[i]$. However, the estimator M assumes that when a failure occurs along Ps, all the connections that route over the failed link would indeed reroute onto link i and is, therefore, a crude estimate of the restoration bandwidth needed due to link failures along Ps. In fact, to minimize restoration bandwidth we need to spread the restoration routes around to share spare bandwidth scattered throughout the network. Minimizing restoration bandwidth is an inherently combinatoric problem. This approach may overestimate the bandwidth that needs to be reserved on some links. Given accurate information of the additional restoration bandwidth that needs to be reserved on each link, it is possible to choose a restoration path that better minimizes $\Sigma R[i]$.

This leads to our new algorithm: *Full Information Restoration* (FIR.) The basic idea is as follows. After selecting the service path Ps, the source node collects the array $T[i]$, $i \in E$, where $T[i]$ is the maximum bandwidth needed on link i if any of the links along Ps fails. This computation is based on the network state before the new restoration connection is routed. The source node assigns a weight to each link in the network:

$$w[i]=\begin{cases}\min(b, T[i]+b-R[i])\cdot W[i] & \text{if } T[i]+b-R[i]>0 \\ & \quad \text{and } i\notin Ps \\ \varepsilon & \text{if } T[i]+b-R[i]\leq 0 \\ & \quad \text{and } i\notin Ps \\ \infty & \text{if } i \in Ps.\end{cases}$$

Then Dijkstra's algorithm is used to select the restoration path Pr using these new weights. In the next section, we present a simple and distributed approach to collect the array $T[i]$ during signaling exchanges, without flooding link state information.

All three algorithms may encounter the so-called trap topology problem [16], [19]. In a trap topology, two link/node disjoint paths exist between the source and destination nodes, but it is possible that none of the algorithms can find a restoration path. The problem arises because of the two-step nature of the three algorithms, each of which selects the shortest service path without considering the goal of subsequently selecting a link/node disjoint restoration path. A min-cost max-flow algorithm may be used to avoid this dilemma. However, since trap topologies are fairly rare in real networks, we do not consider them further in this paper. Interested readers may refer to [16] and [19] for more information.

III. Signaling Procedures for FIR

The FIR algorithm requires the additional link state information in the array $T[i]$ about the bandwidth that needs to be reserved on each link i when any of the links along the service path fails. In this section, we propose the signaling protocol extensions and describe how the additional information is collected as well as how all of the necessary state information is maintained. An early prototype implementation is reported in [13].

As described in Section II-D, the master node of link k along a restoration path keeps track of the bandwidth on link k that has been reserved to protect against failures of any other link i. This information is maintained in $Failother(k)[i]$. In addition, assume that the master node of link k along the service path also keeps track of the bandwidth that has been reserved on other network links to protect against the failure of link k itself. This information is maintained in another local array, called $Failself(k)$, where $Failself(k)[i]$ stores the bandwidth required on link i to restore all connections affected if link k fails. Both $Failself(k)$ and $Failother(k)$ are updated during the signaling process. During the lifecycle of a connection, the operations include creation and deletion. We describe proposed signaling extensions and procedures for updating the local data structures during connection creation and deletion below.

A. Connection Creation

When the source node receives a connection request, it computes a service path Ps using Dijkstra's algorithm with administrative weights based on its network topology database. In this paper, we assume the use of the RSVP Traffic Engineering (RSVP-TE) extensions [8] which allow signaling messages to be explicitly routed from the source to the destina-

tion node. Similar extensions are required for constraint-based routing label distribution protocol (CR-LDP) [9].

Connection creation involves an initial signaling message from the source node to the destination node to set up the service path,[1] then an acknowledgment is returned from the destination node to the source node to finish the service path creation. The acknowledgment signaling message collects the information of the array T used in FIR restoration path selection. After the restoration path selection, the source node then issues another signaling message to the destination node along the restoration path to reserve the restoration resources. At the same time, the source node sends a fourth signaling message along the service path to update the array $Failself(k)[i]$ for link k, $k \in Ps$.

Specifically, after selecting the service path Ps, the source node sends an RSVP-TE PATH message to the destination node along the service path Ps. Upon receiving this PATH message, the destination node sends a RESV message back to the source node traversing Ps in reverse direction. The RESV message carries the $T[i]$ array with i over all links in the network. The destination node initializes the elements of $T[i]$ to zero. As the RESV is processed at the master node for each link k along the service path, $T[i]$ is updated as follows. $T[i] \leftarrow max(T[i], Failself(k)[i])$. This can be done because we assume that each master node maintains the local array $Failself(k)$ for its link k. When the destination node receives the RESV message, it computes the restoration path Pr using the FIR algorithm as discussed in Section II. At this point, the source node sends two PATH messages to the destination node: one is sent along the restoration path to update the parameters for shared restoration resources and the other is sent along the service path to update local link state information. The PATH message sent along the restoration path lists the links on the service path. Recall that restoration bandwidth is not dedicated to a particular connection, but rather shared among restoration paths over nonsimultaneous failures. The master node of link k along the restoration path updates the $Failother(k)$ array as follows: $Failother(k)[i] \leftarrow Failother(k)[i] + b$ if link i is on the service path and b is the requested bandwidth of the connection. The PATH message sent along the service path lists the links on the restoration path. The master node of link k along the service path updates the $Failself(k)$ array as follows: $Failself(k)[i] \leftarrow Failself(k)[i] + b$ if link i is on the restoration path.

The implementation of this approach requires a consistent assignment of network link IDs by all network nodes. Since link state routing gives each node a consistent view of the network topology, each link can be indexed by hashing the node IDs of the two nodes adjacent to each link in a standardized way. Another possible solution is for the network provider to provision a globally unique link ID for each link. A second issue relates to the protocol message size. For large networks, the size of array T may exceed the size of a single packet. Assuming reasonable network diameter and an efficient encoding of the array T, we do not anticipate a significant problem. The implementation details are left for further study.

[1]If multiple connection creation requests arrive simultaneously at each end node of a bidirectional link, there may be contention for the same resources on that link. If this occurs, any failed request is assumed to "crank back" to its source node, where a new setup request is attempted along an alternate service path.

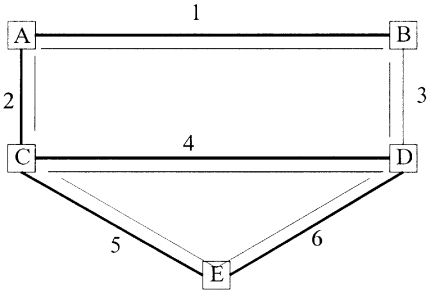


Fig. 2. Example of LSP creation.

B. Example

We consider a simple GMPLS network with five nodes and six links. The network topology, node names, and link IDs are shown in Fig. 2. Suppose there exists a connection of one unit of bandwidth from node C to D with restoration path C-E-D. Assume node A receives a restorable connection request of one unit of bandwidth unit from A to B. At this time, node A has the network topology information including reservation state as follows: $R[1]=R[2]=R[3]=R[4]=0$ and $R[5]=R[6]=1$.

Node A first computes the service path $Ps = \{\mathrm{A}-\mathrm{B}\}$. Then a PATH message is sent along A-B. When B receives the PATH message, it sends out a RESV message along the reverse path from B to A and this RESV message collects the T array. After A receives the RESV message with array T, whose elements are equal to zero in this example, it computes the restoration path using the FIR algorithm, which is $Pr = \{\mathrm{A}-\mathrm{C}-\mathrm{E}-\mathrm{D}-\mathrm{B}\}$ instead of the shortest path $\{\mathrm{A}-\mathrm{C}-\mathrm{D}-\mathrm{B}\}$. Next, A sends two PATH messages from A to B. One message is forwarded along A-B containing information about the links in Pr, which are $\{2,5,6,3\}$. Each master node of the links of Ps updates the $Failself$ array. The second message is forwarded along A-C-E-D-B containing information about the links of Ps (link ID $= 1$). Each master node of links along Pr updates the $Failother$ array correspondingly. In addition, each node updates the available and reserved bandwidth and these changes for the affected links are disseminated to other nodes via OSPF or IS-IS.

C. Connection Deletion

When a client sends a connection deletion request, the source node must delete the connection along the service path and release the reserved bandwidth along the restoration path. Because the $Failself$ and $Failother$ arrays are updated in a distributed fashion, the deletion process needs to update these arrays. In RSVP-TE, the source node sends two PATHTEAR messages to the destination node. One PATHTEAR message is sent along the service path Ps and the other message is sent along the restoration path Pr. The master nodes of link k along the restoration path update the $Failother(k)$ array as follows: $Failother(k)[i] \leftarrow Failother(k)[i] - b$ if link i is on the service path, where b is the required connection bandwidth. Similarly, the master nodes of link k along

the service path update the $Failself(k)$ array as follows: $Failself(k)[i] \leftarrow Failself(k)[i] - b$ if link i is on the restoration path. Updates to $A[i]$ and $R[i]$ are handled similarly as in connection creation.

IV. Node and Fiber Span Failure Protection

The procedures discussed in Section III only consider single link failures. Service providers may also want to protect against two other classes of failure: (single) node failures and fiber span failures. In this section, we will show how to extend the shared mesh restoration approach to handle single node failures and fiber span failures.

The difference between single link and single node failure is that a single node failure causes multiple link failures. To protect against single node failure, $Failother(k)$ maintains an array of network nodes (rather than links), where $Failother(k)[i]$ is the amount of bandwidth needed on link k if node i fails. To handle both link and node failures, the $Failother(k)$ array needs to be indexed by both link and node IDs. This array is updated in a distributed fashion along each restoration path. To convert the SPR algorithm to protect against node failure, we simply find a node-disjoint path instead of a link-disjoint path. For PIR, we need to modify the definition of the parameter $M = max(N(i))$ to take the maximum of all nodes i along the service path, where $N(i)$ is the sum of the service bandwidths of all the links terminating at node i. We then use the PIR algorithm to set the link weight and Dijkstra's algorithm to compute the node-disjoint restoration path. For the FIR scheme, we also need to define the $Failself(k)$ array where $Failself(k)[i]$ is the amount of bandwidth required on link i if node k fails. Also for FIR, we collect the array T among the nodes along the service path. The rest of the algorithm is exactly the same as in the single link failure case except that the restoration path is selected to be node disjoint instead of link disjoint. Thus, the algorithms can all be generalized in the obvious way to handle both link and node failures.

To understand fiber span failure protection, we first describe the optical network model. A ***fiber span*** is the collection of all optical fibers that are co-located in the same cable, conduit, or substructure between two consecutive points of access (such as a manhole, central office, or amplifier site). Multiple optical connections (links) may share a common fiber span, and a single fiber span cut may cause multiple optical links to fail.

The optical network is built on top of the physical fiber span network. Fig. 3 shows an example fiber span network (lower part) and optical network (upper part). In the fiber span network, there are eight nodes and nine fiber spans. In the fiber span network, six nodes are deployed with OXCs at the nodes shown as open circles; the closed circles represent fiber span terminations without an OXC. In the optical network, there are ten optical links, which are routed on top of the fiber span network. To select a restoration path, it is important to know the routing of optical links over fiber spans. For example, suppose the optical link from A to B is routed over the fiber spans A-C, C-a, and a-B. If the service path for a connection from A to C routes directly over the optical link from A-C, then it appears

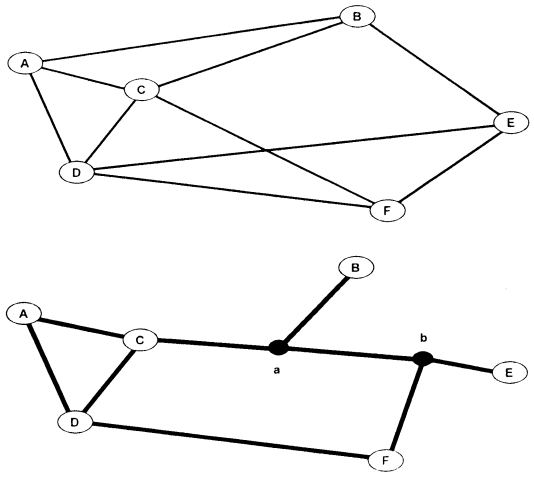


Fig. 3. Optical network model.

from the upper layer optical link map that the restoration path A-B-C would be adequate. However, upon examination of the paths of these optical links on the fiber span layer, it is clear this path is not diverse from the service path. Thus, to update the inventory needed to calculate the restoration paths, we need to allocate management of the fiber span and optical link data among the OXCs in an unambiguous way.

We modify the FIR scheme slightly to support fiber span failure protection as follows. For each fiber span and each optical link, we configure one OXC as the master node of its inventory. The master node is responsible for maintaining link and span information. Specifically, the master node for span s keeps track of the bandwidth $Failself(s)[i]$ that has been reserved on network link i to protect against the failure of span s. Similarly, the master node for link k along a restoration path keeps track of the bandwidth $Failother(k)[i]$ on link k that has been reserved to protect against failures of any span i. The FIR signaling procedures along the service path must now update the span master nodes to correctly maintain the $Failself(s)$ array. Note that a fiber span may not have an OXC present at one or both ends of the span and the mapping from fiber span to master node must be available to the other OXCs in the network. These assumptions are inherent in any shared mesh restoration scheme with disjoint fiber spans.

One issue with this approach is that there is no guarantee that all of the span master nodes will be present along the sequence of links in the optical network layer. One solution requires signaling messages to be explicitly routed through the span master nodes. Details will be discussed in future work.

V. Study Methodology

A. Network and Demands

A 78-node 128-link network representative of a typical intercity backbone network was used as input to a simulation study. The city-to-city demands in this study are generated using nodal densities from a demand forecast. The same network and demand forecast were used in the studies in [2]–[4]. We model our demand distribution on the private line demand that a large

intercity backbone might experience. The source of this demand is almost solely links between IP routers or data switches. Attempts to obtain precise city-to-city forecasts for these private line services are difficult. Instead, we project the cumulative demand at each node, i.e., all traffic terminating at the node. Because nodal demand is more highly aggregated, the coefficient of variation of the difference between projected and actual nodal demands is much less than with city-to-city demands. We also assume that the probability that the source and the destination of a randomly chosen demand in the network is $p(a,b) = p(a)p(b)$ where a, b are two nodes and $p(a)$ is the probability that the source or the destination of a random demand terminates at node a. The bandwidth of each demand is assumed to be one unit, which represents one OC-3, OC-12, or OC-48, etc. For each data point, we generate 100 random runs and calculate the average value for each of the figures of merit of interest, to get reasonable confidence intervals on the results.

B. Figures of Merit

A frequently used figure of merit is the bandwidth consumed over all links in the network. This can be formalized as follows. As before, let $S[i]$ and $R[i]$ be the total bandwidth used for service paths and reserved for restoration paths, respectively, on link i. We define $\alpha = \Sigma S[i]$, $\beta = \Sigma R[i]$. The total bandwidth usage equals $\alpha + \beta$. We measure the ratio of restoration bandwidth to service bandwidth $\Lambda = \beta/\alpha$, often referred to as *restoration overbuild*, which tells us how much extra bandwidth is needed to meet the network restoration objective for single failure protection.

We assume that our network corresponds to an optical transport layer. This layer consists of OXCs connected by high-bandwidth links or wavelengths that are in turn transported over optical transport systems using wavelength division multiplexing (WDM). Costs can then be expected to scale roughly in proportion to the number of OXC ports and to the total optical transport system mileage. The number of OXC ports is directly related to the total bandwidth required on the transport link while the optical transport system mileage is reflective of WDM terminals and optical amplifiers. To capture this, we use the total bandwidth–mileage product $\delta = \Sigma S(i)L(i)$ and $\phi = \Sigma R(i)L(i)$, where $L(i)$ is the length in miles of link i. Since we are particularly interested in the ratio of total restoration bandwidth–mileage product to total service bandwidth–mileage product, we define another figure of merit: $\Pi = \phi/\delta$. In the following, this figure of merit is referred to as the restoration overbuild based on bandwidth–mileage product.

Both the Λ and Π figures of merit assume that each link has infinite bandwidth. In real networks, link bandwidth is limited. When network load is high, some connection requests may be rejected due to the lack of bandwidth for either service path or restoration path. The number of rejected connection requests is another figure of merit often used to measure the network performance.

VI. Simulation Results

Unless otherwise stated, all the data shown is based on single link failures only and is averaged over 100 simulation runs. The results are within 5% variance.

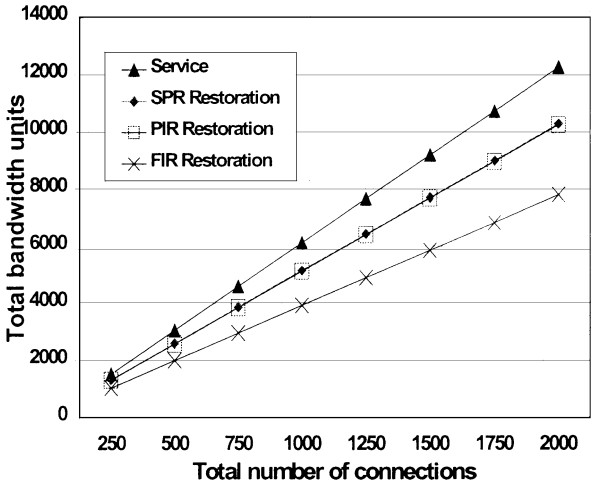


Fig. 4. Total bandwidth.

A. Network Loading Simulations

For this set of experiments, the link capacities are set to infinity. Connection requests arrive one at a time to the network as described previously. We first evaluate the performance based on the total bandwidth usage. Service path selection uses Dijkstra's algorithm. Restoration path selection uses three different algorithms, namely, SPR, PIR, and FIR. Since the bandwidth is infinite, there is no rejection of connection requests. The objective, therefore, is to compare how much restoration bandwidth needs to be reserved in each of the three algorithms to protect all connections against single link failure.

Each algorithm is compared with the same number of total connection demands from 250 to 2000 in increments of 250. Since the service path is always computed by Dijkstra's algorithm in all three scenarios, the total bandwidth for service paths will be the same. Fig. 4 shows the simulation results of total number of connections versus total bandwidth units required for service paths and restoration paths depending on the three different algorithms. All three algorithms use an administrative weight $W[i]$ of one, that is, path weight is equivalent to hop count. The top line is the total bandwidth for service paths. The bottom line is the total bandwidth for restoration paths using the FIR algorithm. In the middle, the two lines giving the total bandwidth for restoration paths using SPR and PIR overlap. In this simulation, we see that shared restoration requires much less bandwidth for restoration than for service in all three algorithms. Among the algorithms, we notice that FIR requires substantially less restoration bandwidth than either SPR or PIR, and there is little difference between SPR and PIR (basically, PIR performs no better than shortest path). Fig. 5 shows the required restoration bandwidth divided by the service bandwidth for each of the three algorithms (that is, the restoration overbuild).

It is clear that the more connections, the smaller the restoration overbuild that is required. This is because more connections provide more opportunity for restoration bandwidth sharing. The restoration overbuilds for SPR and PIR range from 0.83 to 0.88, while for FIR, they range from 0.63 to 0.68, which is a significant reduction in overbuild capacity. Also, note that PIR performs marginally better than SPR; the overbuild ratio flattens out after 1000 connections.

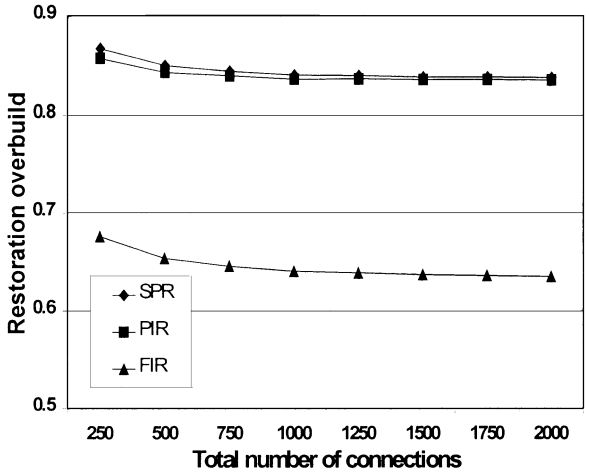


Fig. 5. Restoration overbuild.

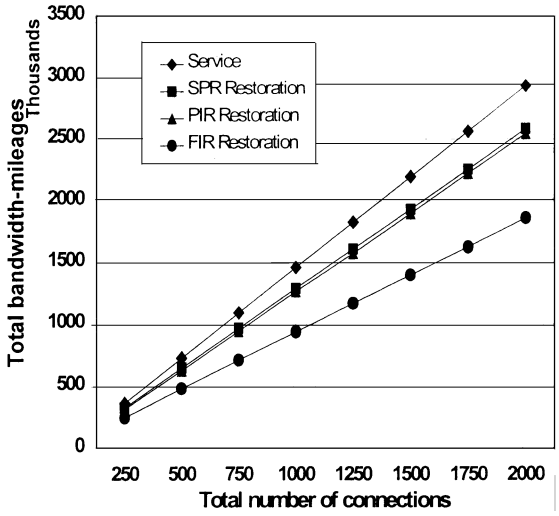


Fig. 6. Total bandwidth–mileage product.

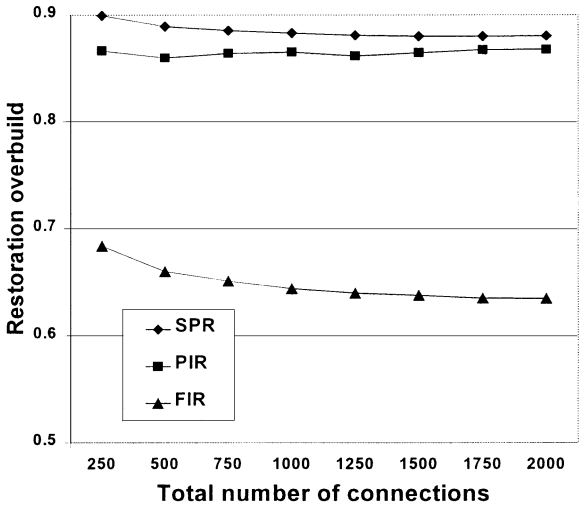


Fig. 7. Overbuild with bandwidth–mileage product.

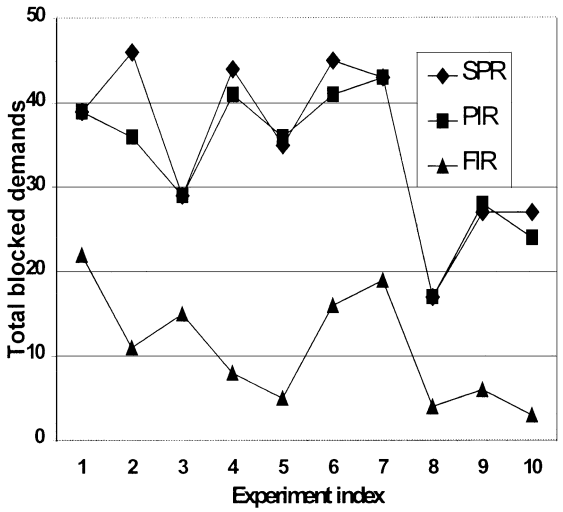


Fig. 8. Rejected requests (admin weight = 1).

Figs. 6 and 7 consider the bandwidth–mileage product figure of merit. The administrative weight for each link is set to its mileage. These simulations show that runs with a figure of merit of bandwidth usage and administrative weight equal to one give similar results to those with a figure of merit of bandwidth–mileage product and administrative weight equal to link mileage. However, Fig. 7 shows that the restoration overbuild is higher using the mileage-based administrative weight than with the bandwidth-based administrative weight (Fig. 5) for both the SPR and PIR algorithms. FIR performs almost the same in this case as when an administrative weight of one is used, while PIR obviously performs better than SPR in this case. We arrived at similar observations when we evaluated a slightly sparser network with 115 nodes and 128 links.

B. Simulations With Rejected Connection Requests

The second set of experiments study the behavior of the algorithms with respect to the number of rejected connection requests when the network is overloaded. In these experiments, we set each link bandwidth to 100 units of bandwidth (channels). Demands are chosen as described previously. As these connection requests are routed, we keep track of how much service bandwidth is used per link, as well as how much restoration bandwidth is needed on each link for each link failure (similar to the $Failother$ array). If there is a lack of available bandwidth when selecting either the service path or the restoration path, then the connection request is rejected. We performed ten experiments with different random number generator seeds. Figs. 8 and 9 show the numbers of requests rejected after 500 connections are loaded on to the network using different restoration algorithms. Fig. 8 is the result of path selection with administrative weights set to one (resulting in hop count = path weight) and Fig. 9 is the result of path selection with the administrative weights set to link mileage. Note that for the PIR and FIR methods, the link weights are defined by the parameter $w[i]$ from the previous equations.

Figs. 8 and 9 both show that FIR performs better than SPR and PIR for either choice of administrative weight. PIR is slightly better than SPR, but the difference is marginal. Comparing the different weights used for path selection, it seems that using the link mileage product as the weight has a slightly higher rejection probability than hop count. This observation needs further study.

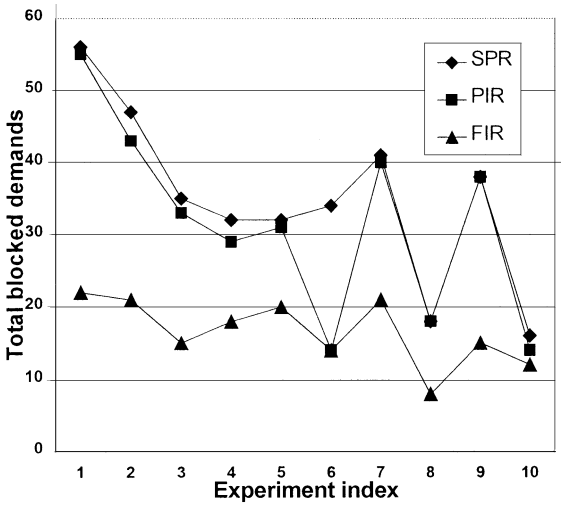


Fig. 9. Rejected requests (admin weight = mileage).

VII. Possible Variations

In the shared mesh restoration scheme proposed above, a restoration path that is disjoint from the service path is precomputed. The restoration path shares a pool of restoration bandwidth on its links. Since resources are reserved but not allocated, they can be shared by restoration connections that are protecting path-disjoint service connections. Upon detecting or being notified of a failure, restoration messages are sent along the restoration path to establish the restoration connection. Nodes allocate resources (e.g., time slot, wavelength, etc.) dynamically at the time of restoration connection creation. Thus, bandwidth can be shared across multiple independent failures. However, the time required to reroute a connection to its restoration path after a failure has occurred (the ***restoration time***) is strongly related to the number of nodes on the restoration path. In this section, we describe two potential variations to the shared mesh restoration scheme to further increase the amount of bandwidth sharing and reduce the restoration time. We do not propose detailed methods here for these variations, but rather, for completeness, give high level descriptions. The first variation to reduce the restoration time is to decrease the number of nodes on the restoration path. To provide 100% restoration after any single failure, two restoration paths cannot share resources on common links if their service paths may be subject to a common failure. However, if the service provider only protects against single link failures and each network node is able to perform failure detection and perform restoration, segment-based restoration may both increase the bandwidth resource sharing and decrease the number of restoration path cross-connects. Methods have appeared in the literature for segment-based restoration, for example, see [26].

Fig. 10 shows a simple example network with nine nodes and eleven links. The solid lines are the service paths and the dotted lines are the restoration paths. There are two service connections: connection S1 from node 1 to node 9 and connection S2 from node 6 to node 9. S1 has restoration path R1 equal to 1-4-5-3-8-9 and S2 has restoration path R2 equal to 6-4-5-3-8-9. If we attempt to do end-to-end shared mesh restoration, R1 and

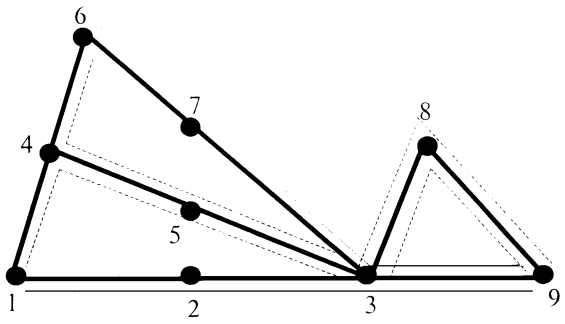


Fig. 10. Segment sharing example.

R2 cannot share bandwidth since S1 and S2 will fail simultaneously if link 3-9 fails. However, if node 3 is able to detect connection failure and perform restoration, we can separate S1 into two segments: S11 consisting of nodes 1-2-3 and S12 consisting of nodes 3-9. Similarly, S2 can be partitioned into S21 consisting of nodes 6-7-3 and S22 consisting of nodes 3-9. If we treat each segment as an independent connection, the restoration paths for S11 and S21 can share the bandwidth reserved on common links. At the same time, the number of nodes on the restoration paths is reduced.

If the shortest path is selected as the service path for each connection, the efficiency of bandwidth sharing on the total restoration bandwidth is completely dependent on the restoration path selection algorithm. No matter what restoration path selection algorithm is used, end-to-end shared mesh restoration can be extended to adopt this variation, referred to as *segment-based* shared mesh restoration. The only changes from basic shared mesh restoration are to select the restoration nodes at segment ends and to update the node data structures appropriately. If the service path and the selected restoration path have any nodes in common, each common node is selected as a restoration node. Thus, the original end-to-end service connection is partitioned into several segments, and this connection is protected segment by segment. Restoration is handled within each segment, and each segment is treated as an individual connection to update the related node data structures. This variation shortens the length of restoration paths and provide more chances for bandwidth sharing, as shown in Fig. 10. Of course, this variation does not protect against failure of the restoration nodes.

Another variation is *diverse service path selection*. Usually, the service path is selected to be the shortest path. If two connections between the same $\langle$source, destination$\rangle$ follow the same service path, we obtain no sharing of restoration bandwidth between the two connections. However, if the two connections use diverse service paths, then they can share the same restoration bandwidth.

Fig. 11 shows a simple example with a network of four nodes and five links with two connections from node 1 to node 2. The solid lines are the service paths and dotted lines are the restoration paths. In Fig. 11(a), without link-diverse service paths, six units of bandwidth are needed, while in Fig. 11(b), with link-diverse service paths, only five units of bandwidth are needed, assuming each connection consumes one unit of bandwidth. How does one automatically select diverse service paths in a distributed way using the information that is available at the source

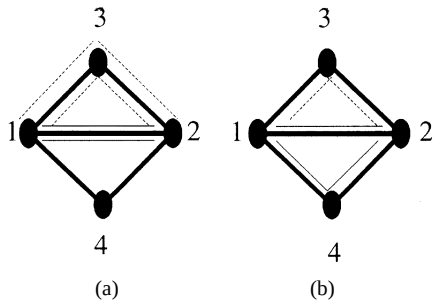


Fig. 11. Diverse path selection example.

node? We assume that the network is running a link-state routing protocol, such as OSPF. Thus, each node has network topology and link resource information. Each connection request arrives in sequence. The node receiving the connection request computes the service path and restoration path. Thus, each node has complete information for all service connections that originated from this node. To automatically select diverse service paths, one may use a simple heuristic for selecting the service path by computing the shortest path using modified link weights. For example, each node may keep a local link weight array $w[i]$, which is initially set equal to the administrative weight $W[i]$. After a service path is selected, the link weights at this node are updated as follows: $w[i] \leftarrow w[i] + \alpha W[i]$ if link i is in the service path, otherwise $w[i]$ remain unchanged, where α is a value between 0 and 1. By increasing the weight on links along the service path for a connection, we steer subsequent connections away from those links. When service connections are deleted, the link weights must be adjusted accordingly. Note that the value of α can be tuned, depending on the network topology and expected traffic patterns.

We have studied several other alternatives for improving distributed shared mesh restoration, such as releasing service bandwidth after connection failure, which allows this bandwidth to be used for restoration and reduces the restoration overbuild. Further study is required to evaluate the different alternatives to shared mesh restoration against the metrics of restoration overbuild and failure recovery time.

VIII. Conclusion

We have analyzed the problem of distributed restoration path selection for restorable connections in a GMPLS shared mesh restoration architecture. In particular, we propose a new restoration path selection algorithm, called *Full Information Restoration*, that uses signaling protocol extensions to distribute and collect additional link state information. This method uses bandwidth most efficiently among the algorithms studied, since it keeps accurate information about the amount of reserved bandwidth that must be reserved on each link in the network to restore any single link failure. The FIR method is based on efficient distributed routing and signaling protocols and does not rely on previous centralized approaches. We compare the FIR method with two other well-known methods that approximate the network state information, namely, the Shortest Path Restoration and Partial Information Restoration algorithms. The PIR algorithm shows little advantage over the SPR algorithm. However, the FIR method is shown to give significant bandwidth advantages over both. We conclude that the savings in restoration bandwidth from shared restoration mesh is significant, especially when using FIR, and that considerable savings could be obtained by extending the GMPLS signaling protocol to support the FIR algorithm. Future work will focus on joint distributed service path and restoration path selection algorithms to further improve the bandwidth resource utilization.

References

[1] M. Kodaliam and T. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proc. IEEE INFOCOM*, 2000, pp. 902–911.

[2] J. Strand, R. Doverspike, and G. Li, "Importance of wavelength conversion in an optical network," *Opt. Networks Mag.*, vol. 2, pp. 33–44, May/June 2001.

[3] R. Doverspike, G. Sahin, J. Strand, and R. Tkach, "Fast restoration in mesh network of optical cross-connects," in *Proc. Optical Fiber Communications Conf.*, Mar. 1999, pp. 170–172.

[4] R. Doverspike, J. Strand, and G. Li, "Analysis of restoration against node failure in a mesh network of optical cross-connects," presented at the 5th INFORMS Telecommunications Conf., Boca Raton, FL, Mar. 2000.

[5] E. Rosen *et al.*, "Multiprotocol label switching architecture," IETF, RFC 3031, Jan. 2001.

[6] D. Awduche *et al.*, "RSVP-TE extensions to RSVP for LSP tunnels," Network Working Group, RFC 3209, Dec. 2001.

[7] P. Ashwood-Smith *et al.*, "Generalized MPLS—Signaling functional description," IETF, Internet draft, work in progress, Nov. 2001.

[8] L. Berger *et al.*, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) extensions," IETF, RFC 3473, Jan. 2003.

[9] P. Ashwood-Smith *et al.*, "GeneralizedMulti-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) extensions," IETF, RFC 3472, Jan. 2003.

[10] K. Kompella *et al.*, "OSPF extensions in support of generalized MPLS," IETF, Internet draft, work in progress, Feb. 2001.

[11] K. Kompella *et al.*, "IS-IS extensions in support of generalized MPLS," IETF, Internet draft, work in progress, Feb. 2001.

[12] S. Kini *et al.*, "Shared backup label switched path restoration," IETF, Internet draft, work in progress, May 2001.

[13] G. Li, J. Yates, R. Doverspike, and D. Wang, "Experiments in fast restoration using GMPLS in optical/electronic mesh networks," presented at the Optical Fiber Communications Conf., 2001, postdeadline paper.

[14] R. Doverspike and J. Yates, "Challenges for MPLS in optical network restoration," *IEEE Commun. Mag.*, vol. 39, pp. 89–96, Feb. 2001.

[15] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal space capacity allocation by successive survivable routing," in *Proc. IEEE INFOCOM*, 2001, pp. 699–708.

[16] G. Li, R. Doverspike, and C. Kalmanek, "Fiber span failure protection in optical networks," *Opt. Networks Mag.*, vol. 3, pp. 21–31, May/June 2003.

[17] G. Li *et al.*, "RSVP-TE extensions for shared-mesh restoration in transport networks," IETF, Internet draft, work in progress, July 2001.

[18] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Norwood, MA: Kluwer, 1999.

[19] D. Dunn, W. Grover, and M. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE J. Select. Areas Commun.*, vol. 2, pp. 88–89, Jan. 1994.

[20] B. Caenegem, W. Parys, F. Turck, and P. Demeester, "Dimensioning of survivable WDM networks," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1146–1157, July 1998.

[21] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks, Part I—Protection," in *Proc. IEEE INFOCOM*, 1999, pp. 744–751.

[22] C. Qiao and D. Xu, "Distributed partial information management (DPIM) schemes for survivable networks—part I," in *Proc. IEEE INFOCOM*, June 2002, pp. 302–311.

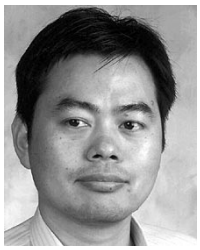
[23] S. Datta *et al.*, "Efficient channel reservation for backup paths in optical mesh networks," in *Proc. GLOBECOM*, vol. 4, Nov. 2001, pp. 2104–2108.

[24] E. Bouillet *et al.*, "Enhanced algorithm cost model to control tradeoffs in provisioning shared mesh restored lightpaths," in *Proc. Optical Fiber Communications Conf.*, Mar. 2002, pp. 544–546.

[25] S. Chaudhuri, "Comparison of centralized and distributed provisioning of lightpaths in mesh optical networks," in *Proc. Optical Fiber Communications Conf.*, Anaheim, CA, Mar. 2001, p. MH4 (1–3).

[26] M. Deng, D. F. Lynch, S. J. Phillips, and J. R. Westbrook, "Algorithms for restoration planning in a telecommunications network," presented at the Workshop on Algorithm Engineering and Experimentation (ALENEX'99), Baltimore, MD, 1999.

[27] E. Bouillet, J. Labourdette, G. Ellinas, R. Ramamurthy, and S. Chaudhuri, "Stochastic approaches to compute shared mesh restored lightpaths in optical network architectures," in *Proc. IEEE INFOCOM*, June 2002, pp. 801–807.



Guangzhi Li (M'00) received the M.S. and Ph.D. degrees in computer science from the College of William and Mary, Williamsburg, VA.

He is a Senior Technical Staff Member with AT&T Research Labs, Florham Park, NJ. He has published more than 30 technical papers in journals and conferences as well as several contributions to IETF/OIF/ITU standardization organizations. His research interests include IP-based control plane for optical networks, optical layer restoration/protection schemes and algorithms, network simulation and performance evaluations.

Dongmei Wang (M'01) received the Ph.D. degree from the College of William and Mary, Williamsburg, VA, in 2000.

She is a Senior Technical Staff Member with AT&T Research Labs, Florham Park, NJ. Currently, she is engaged in IP-based control plane for optical network project, especially the routing protocol extension and network-to-network interworking, and a resilient packet ring (RPR) network project, focusing on MSP technology evaluation, RPR fairness algorithm and performance simulation. Her research interests include network architecture, protocols, and simulations.

Charles Kalmanek (M'93) received the B.S. degree in applied physics from Cornell University, Ithaca, NY, in 1980, the M.S. degree in electrical engineering from Columbia University, New York, in 1981, and the M.S. degree in computer science from New York University, New York, in 1988.

He joined AT&T Bell Labs in 1980, has managed advanced development and research groups since 1989, and is currently Manager of the Networking Research Division in AT&T Labs. He has worked in a broad range of areas including cable access, IP telephony, routing and switching. He is currently working on metro access networks, network measurement, and content distribution. His research interests include network architecture, protocols, and systems.

Robert Doverspike (M'92–SM'97) received the undergraduate degree from the University of Colorado, Boulder, and the M.S. and Ph.D. degrees in mathematics from Rensselaer Polytechnic University, Troy, NY.

He started with Bell Labs in 1979 and, upon divestiture of the Bell System, joined Bellcore (now Telcordia). In 1997, he joined AT&T Research Labs, where he currently manages the Transport Network Evolution Research Department. He has extensive experience with optimization of metro networks and, in particular, optimization of multilayered transmission and switching networks. His current work includes advanced transport and IP network architectures, network restoration methods for optical cross-connects (for which he has numerous patents), and methods for IP over optical-layer integration. He has published in journals on telecommunications, optical networks, mathematical programming, operations research, applied probability, and network management.

Dr. Doverspike is a member of the Mathematical Programming Society, the Optical Society of America (OSA), and INFORMS, and serves as Associate Editor of the *Heuristics Journal* and the *Operations Research Journal*. He has held key leadership positions of multiple international telecommunications societies and conferences, most prominently the INFORMS Technical Section on Telecommunications.