Restorable Dynamic Quality of Service Routing

Murali Kodialam and T. V. Lakshman, Lucent Technologies

ABSTRACT

The focus of quality-of-service routing has been on the routing of a single path satisfying specified QoS constraints. Upon failure of a node or link on the path, a new path satisfying the constraints has to be established. However, resources needed to satisfy the QoS requirements are not guaranteed to be available at the rerouting instant, so QoS is not guaranteed upon failure. Restorable QoS routing, where active and backup paths must be simultaneously set up, has only recently been studied. This is mostly motivated by the incorporation of mechanisms to establish QoS guaranteed paths with failure protection in multiprotocol label switching networks. This article describes some recently developed algorithms for dynamic routing of restorable QoS guaranteed paths.

INTRODUCTION

The objective of quality of service (QoS) routing has been to select a single path that satisfies the specified QoS requirements and optimizes network usage. This topic has been extensively studied [1–4]. Path selection algorithms such as widest shortest path and minimum interference routing are examples. In these schemes, failure resilience is not an objective, and if a link or node fails, all paths that traverse the failed link or node have to be re-routed. Since setting aside resources for failure rerouting was not an objective of the initial routing, there is no guarantee that rerouting can be successfully done, so the initially routed paths are not fault-tolerant or restorable. In circuit-switched networks, restorable routing is commonly used (e.g., setting up paths with 1 + 1 or 1:1 protection). In 1 + 1 protection, two disjoint paths are used and data is sent on both paths. The receiver picks the better path to use and discards data from the other. In 1:1 protection, two disjoint paths also exist, but data is sent only on one active path. The backup path is activated by signaling only if the active path fails.

In the context of packet networks, recently there has been much interest in setting up QoS guaranteed paths that are resilient to faults. This interest stems from the incorporation of mechanisms to support restoration in the path setup signaling for multiprotocol label switching (MPLS) networks. This article discusses recently developed schemes for restorable routing with QoS guarantees. We first describe the different network resource information models pertinent to restorable routing with resource reservations. We then describe different restoration schemes and the algorithms that can be used. We also describe aspects of signaling relevant to restorable QoS routing. The description of schemes is only for the case where the QoS guarantees are bandwidth guarantees, and the two are used synonymously in the rest of the article.

We describe the schemes using the terminology of MPLS networks since the described schemes are most likely to be used in MPLS networks. At the ingress points of an MPLS network, incoming packets are encapsulated with labels that are used to forward the packets along label switched paths (LSPs). These LSPs can be thought of as virtual traffic trunks and are set up using signaling protocols such as Resource Reservation Protocol with Traffic Engineering (RSVP-TE) or constraint routed label distribution protocol (CR-LDP) [5]. One of the goals in setting up LSPs is to permit service providers to traffic engineer their networks, and to dynamically provision QoS or bandwidth guaranteed paths that can be made failure-resilient [6]. Failure resilience is achieved by restoration mechanisms that allow backup paths onto which traffic can be quickly redirected upon failure detection, to be set up simultaneously with the active path. This necessitates extensions to routing and signaling protocols, and the development of new path selection algorithms.

RESTORABLE QoS ROUTING

We assume that all demands (LSP setup requests) are not known a priori. Hence, we are only interested in online (or dynamic) routing that routes LSP requests that arrive one by one. The main issues are the mode of restoration (local vs. end-to-end restoration), the failure modes we protect against, and the information about the network that can be made available to path selection algorithms.

0163-6804/02/$17.00 © 2002 IEEE

Restoration Model

Backup for connections in a network can be built so that there is either *path restoration* or *local restoration*. In path (or end-to-end) restoration, the idea is to provide a backup path from the source to the destination for each active LSP. This backup path is (link or node) disjoint from the active path. However, the drawback with this approach is that when there is a link or node failure, this failure information has to propagate back to the source, which in turn switches traffic from the active to the backup path. This has to be done for all the LSPs that use this link on their active paths. Note that the link failure information has to be potentially propagated to all nodes in the network. Path restoration along with the information transfer on failure is illustrated in Figs. 1 and 2.

The time taken for this information propagation to the source may not be acceptable for many applications. This necessitates the use of a local restoration scheme. In local restoration, the backup paths are set up for every node or link. Therefore, upon failure the first upstream node can locally direct traffic onto the backup path bypassing the failure. These two different backup methods are explained in detail later.

Failure Modes

We consider two different failure modes against which we protect. The first is against *single link failures*. In this case, we have to protect the active path against all single link failures. The second is *single element (node or link) failures*. The amount of resources needed to provide backup for the second mode will be greater. In terms of detecting failures, we assume the following: In the single link failure model, when a link fails, the two nodes that are at the endpoint of the link know that the link has failed, and immediately switch all the demands that go on this link to the alternate path. In the single element failure model, when a node fails we assume that all the link interfaces at that node fail, and therefore all links that are incident on the node fail. This is detected as in the link failure case, and the LSPs are routed across the failed node. Note that in the case of single element failures, there has to be a backup path for all node failures. We do not provide a backup if the source or destination of the traffic fails. Taking care of single node failures almost handles all link failures also except for the last link on the active path. This is illustrated in Fig. 3. Therefore, for the single element failure case we protect against all single node failures and the failure of the last link.

Backup Paths

For path restoration, the backup path has to be link disjoint with the active path when we are protecting against link failures. The node failure case can be transformed to the link failure case, as shown later.

For local restoration to be possible with single link failures, the backup bypass path for a link (i, j) can be any path connecting nodes i and j that does not include link (i, j). This backup path for link (i, j) can include any link, including any links on the active path for the current LSP (apart from link (i, j)), as well as any links that are used in the backup path for other links on the active path for this LSP. For the single element failure case, the backup path for the failure of node k involves doing the following: First determine the links (j, k) and (k, l) in the active path. If node k fails, it will result in the failure of all links incident on node k, in particular link (j, k). Therefore, the failure will be detected at node j, and if there is an alternate path from node j to node l (or some other node between l and the destination t), node j can divert traffic along this backup path. Note that the backup path for the failure of node k has to avoid all links incident on node k.

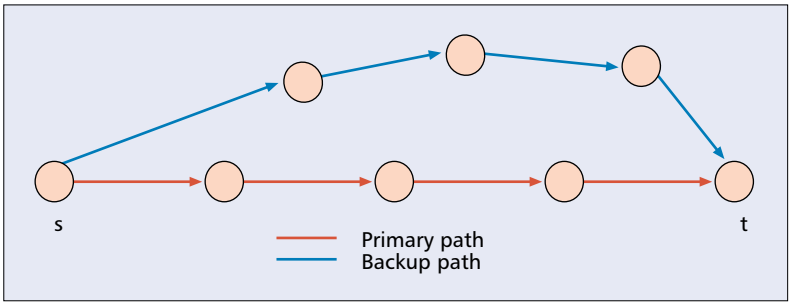

Figure 1. *Path restoration: active and backup paths.*

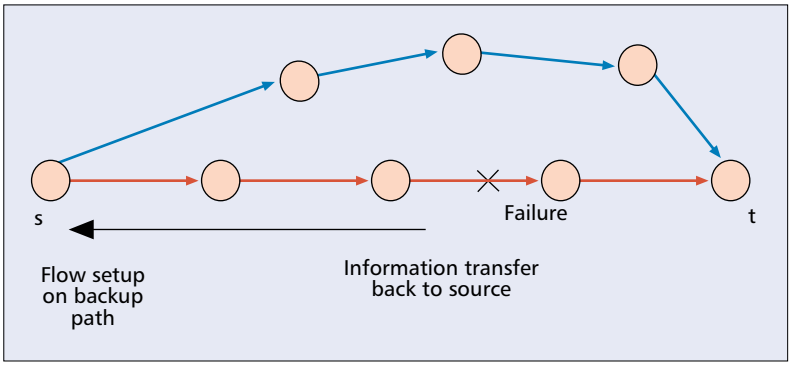

Figure 2. *Information transfer on link failure.*

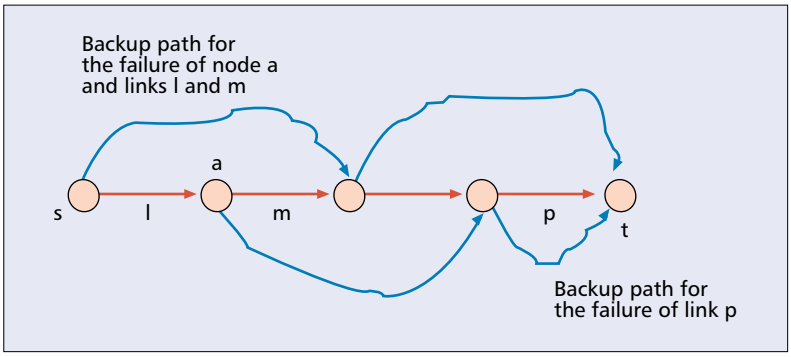

Figure 3. *A backup path for single element failure.*

Sharing Backup Links

Capacity on the active path cannot be shared. Capacity on the backup path can be shared at two levels. *Interdemand sharing* is the case where the backup reservation belonging to different LSPs that do not share links in the active path can be shared. For example, if two LSPs with equal

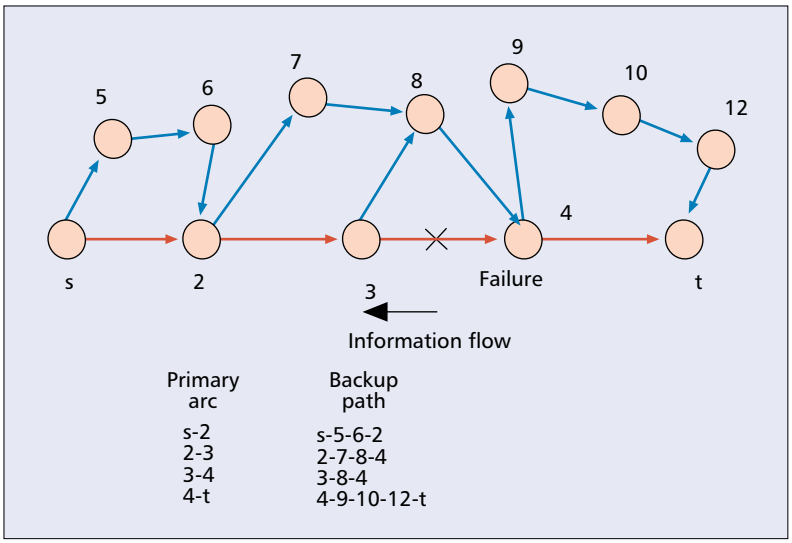

Figure 4. *Active and backup paths for single link failure.*

bandwidth demands between a given source and destination do not have any links in common on their active paths, the backup path for these two LSPs can be shared completely. This is an extreme case. However, even if the two LSPs' active paths have some links in common, it may still be possible to share capacity on the backup path. The second case is *intrademand* sharing. We use Fig. 4 to illustrate intrademand sharing. In Fig. 4 note that link (8, 4) is used to backup links (2, 3) and (3, 4). This in turn means that backup capacity is shared on this link. The algorithms we outline exploit both inter- and intrademand sharing in order to minimize the amount of bandwidth consumed. Figure 5 illustrates sharing backup in the case of single element failure.

The Information Model

The amount of sharing that is possible depends on the kind of link usage information made available to the path selection algorithm. There are three cases of interest.

The first is what we call the *no information* case. Here, the only information made available about the network is the total bandwidth that has been reserved on each link. This information is obtainable from routing protocol extensions. Since the link capacities are all known, the residual bandwidth on all links can be inferred. Note that *no information* about backup resource usage is available since the amount of bandwidth utilized separately by the active and backup paths on a link is not known.

Next is the case where there is *complete information*; that is, the path selection algorithm knows the routes for the active and backup paths of all LSPs currently in progress. This information is obtainable if path selection is done in a centralized manner. The very large amount of information that needs to be sent makes it impractical to disseminate this complete information to all nodes using link state flooding mechanisms.

In the third *partial information* case, the information available to the routing algorithm is slightly more than that in the no information case. The additional information is that for each link, instead of knowing only the total (or equivalently residual) bandwidth usage, we now separately know the total bandwidth used by active paths and the total bandwidth used by backup paths. This incremental information is very useful. It is possible to disseminate it in a distributed manner by incremental additions to proposed traffic engineering extensions to routing protocols.

In the first no information scenario, it is not possible to do any interdemand sharing of the backup paths since the relevant information on backup bandwidth usage is not available. Intrademand sharing is still possible since the source node has the backup bandwidth usage information for the current demand. The second complete information scenario permits the best sharing but is not always practical, so it is mainly useful only for comparison purposes. The third partial information scenario is fairly modest in terms of the amount of information to be maintained. Because only aggregate information is needed and no per-LSP information, it is easy to maintain and use this information in a distributed fashion. Therefore, online routing of bandwidth guaranteed active and backup paths for restoration under the partial information model is the main case of interest. Furthermore, as discussed later, even though the partial information case only provides aggregate (and not per-demand) link usage information, it is still possible to make exact backup bandwidth reservations at each link. This makes the performance of the partial information case very close to that of the ideal complete information case.

Notation and Definitions

We consider a network of n nodes (switches/routers) and m links. All the links are assumed to be directional. We consider the setup request for LSP k to be defined by a triple (o_k, t_k, b_k). The first field, o_k, specifies the ingress router, the second, t_k, specifies the egress router, and the third, b_k, specifies the amount of bandwidth required for LSP k. For each LSP setup request, an *active path* and a *backup path* have to be set up. In the single link failure case, restoration is guaranteed for any single link failure in the network. In the single element failure case, restoration is guaranteed for either a single link or single node failure. If we determine that there is insufficient bandwidth in the network to set up either the active path or the backup path for the current request, this request is rejected.

Requests are assumed to come one at a time. For ease of notation, assume that the current request is for b units of bandwidth between source node s and destination node t. If this request is accepted, we note that all links on its active path will reserve b units of bandwidth for this request.

Let A_{ij} represent the set of LSP active paths that use link (i, j) and B_{ij} represent the set of LSPs that use link (i, j) as a backup. Let F_{ij} represent the total amount of bandwidth reserved for the LSPs that use link (i, j) on the active path. Let G_{ij} represent the total amount of bandwidth reserved by LSPs that use link (i, j) on its backup path.

$$F_{ij} = \sum_{k \in A_{ij}} b_k$$

and

$$G_{ij} = \sum_{k \in B_{ij}} b_k.$$

Let $R_{ij} = C_{ij} - F_{ij} - G_{ij}$ represent the residual bandwidth of link (i, j). In the complete information case we assume that each node knows the sets A_{ij} and B_{ij} for all links (i, j) in the network. In the partial information case we assume that each node knows the value of F_{ij}, G_{ij}, and R_{ij} for all links (i, j) in the network. Since we do not have any knowledge of the requests that will arrive in the future, the objective of the routing algorithm is to determine the active and backup paths for the current request so as to optimize use of the network infrastructure. A reasonable objective then is to *minimize the sum of the bandwidths used by the active and backup paths*. In the case where no restoration is needed (i.e., we just have to determine one path), this objective leads to min-hop routing.

Path Restoration

We first consider path restoration for single link failures under the three information scenarios. For all the cases, we need to set up an active path and a link disjoint backup path. The bandwidth reserved on the active path is b. The backup bandwidth reservation can be less than b due to the possibility of backup sharing. The amount of sharing achieved depends on the information model. The objective used in finding the paths is that the total bandwidth used by the two paths be minimized.

No Information

Since no information is known other than R_{ij}, the path selection algorithm does not know the backup bandwidth currently used on the different links. Hence, it cannot determine whether sharing of backup bandwidth is possible, and must assume that no sharing is possible. Thus, bandwidths of b units have to be reserved on each link in the active as well as the backup path. Clearly, if $R_{ij} < b$ for link (i, j), that link cannot be used for the active or backup path for the current request.

With the objective of minimizing the total amount of bandwidth consumed by both the active and backup paths, we have to determine two link disjoint paths. Since the bandwidth consumed on each link is b units, the objective of minimizing the total amount of bandwidth consumed is equivalent to determining a pair of link disjoint paths, where the total number of links is minimum. This problem can be formulated as a standard network flow problem where each link has unit cost and unit capacity. There is a supply of two units at node s and a demand for two units at d. Any minimum cost flow algorithm can be used to solve this problem. A very fast algorithm for solving this problem is given in [7] and involves solving two shortest path problems.

An alternative would be to route two disjoint paths using the minimum interference objective.

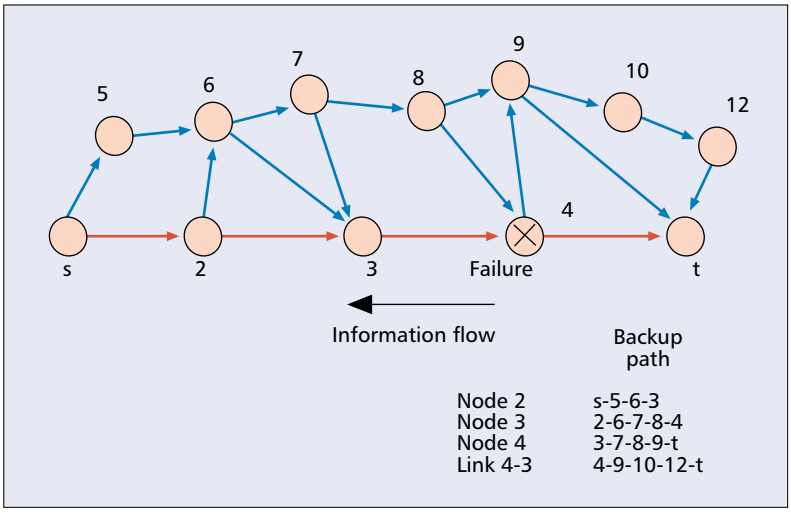

Figure 5. *Active and backup paths for single element failure.*

The minimum interference objective and an algorithm for dynamic routing of nonrestorable bandwidth guaranteed paths using this objective are described in [2]. An algorithm for dynamic routing of restorable bandwidth guaranteed paths using the minimum interference objective is developed in [8]. The no information model gives an upper bound on the bandwidth consumed for the partial information model.

Complete Information

The complete information model, although practical only in a centralized implementation, is nevertheless useful for comparison purposes since it gives a lower bound on bandwidth used in the partial information model. Routing under complete information can be formulated as an integer linear programming problem [9].

In this scenario, the sets $\mathcal{A}_{ij}$ and $\mathcal{B}_{ij}$ are known for all links (i, j). Since we are assuming robustness under single link failures, it is possible to share backup paths between requests whose active paths do not share the same link. To formulate this problem, we first define the quantity θ_{ij}^{uv} for each link pair (i, j) and (u, v). This quantity θ_{ij}^{uv} is the cost (bandwidth usage) of using link (u, v) on the backup path if link (i, j) is used in the active path. To compute the value of θ_{ij}^{uv} we first define the set $\phi_{ij}^{uv} = \mathcal{A}_{ij} \ll \mathcal{B}_{uv}$. This is the set of demands that use link (i, j) on the active path and link (u, v) on the backup path. Let the sum of all the demand values in the set ϕ_{ij}^{uv} be represented by $\delta_{ij}^{uv} = \sum_{k \in uvij}$. Recall that the current demand is for b units of bandwidth between nodes s and d. Now θ_{ij}^{uv} is defined as follows:

$$\theta_{ij}^{uv} = \begin{cases} 0 & \text{if } \delta_{ij}^{uv} + b \le G_{uv} \\ & \text{and } (i,j) \ne (u,v) \\ & \text{if } \delta_{ij}^{uv} + b > G_{uv} \\ \delta_{ij}^{uv} + b - G_{uv} & \text{and } R_{uv} \ge \delta_{ij}^{uv} + b - G_{uv} \\ & \text{and } (i,j) \ne (u,v) \\ \infty & \text{otherwise.} \end{cases}$$

The complete information model, although practical only in a centralized implementation, is nevertheless useful for comparison purposes since it gives a lower bound on bandwidth used in the partial information model.

The motivation for the above is as follows: Since links (i, j) and (u, υ) cannot be on both the active and backup paths, the value of $\theta_{ij}^{u\upsilon}$ is set to infinity if $(i, j) = (u, \upsilon)$. The quantity $\delta_{ij}^{u\upsilon}$ represents the amount of backup capacity on link (u, υ) that cannot be used to back up the current demand, if link (i, j) is used in the active path. This is because $\delta_{ij}^{u\upsilon}$ is the amount of bandwidth needed on link $u\upsilon$ to backup the active paths currently traversing link ij. Therefore, taking the current request into account as well, a total of $\delta_{ij}^{u\upsilon} + b$ units of backup bandwidth are needed on link $u\upsilon$ if the current request were to traverse link ij and use link $u\upsilon$ for backup. Recall that $G_{u\upsilon}$ is the amount of backup (and hence shareable) bandwidth usage currently on link $u\upsilon$. Then the current request can be backed up on link (u, υ) without reserving any additional bandwidth if $\delta_{ij}^{u\upsilon} + b \le G_{u\upsilon}$. Since only $G_{u\upsilon}$ units of bandwidth is shareable, if $\delta_{ij}^{u\upsilon} + b > G_{u\upsilon}$, an additional reservation of $\delta_{ij}^{u\upsilon} + b - G_{u\upsilon}$ units is necessary. If this bandwidth is not available, this backup path is not feasible, so for this request the backup cost of link $u\upsilon$ is set to infinity.

Let vector **x** represent the flow on the active path, where x_{ij} is set to 1 if link (i, j) is used in the active path. Let vector **y** represent the flow on the backup path, where y_{ij} is set to 1 if link (i, j) is used on the backup path. The routing with full information can be formulated as the following integer programming problem with quadratic constraint:

$$\min b \sum_{(i,j)\in E} x_{ij} + \sum_{(u,\upsilon)\in E} z_{u\upsilon}$$

$$\sum_j x_{ij} - \sum_j x_{ij} = \begin{cases} 0 & \text{if } i \ne s,t \\ 1 & \text{if } i = s \\ -1 & \text{if } i = t \end{cases}$$

$$\sum_j y_{ij} - \sum_j y_{ij} = \begin{cases} 0 & \text{if } i \ne s,t \\ 1 & \text{if } i = s \\ -1 & \text{if } i = t \end{cases}$$

$$z_{u\upsilon} \ge \theta_{ij}^{u\upsilon} x_{ij} y_{u\upsilon} \quad \forall(i,j) \quad \forall(u,\upsilon)$$

$$x_{ij} y_{ij} \in \{0,1\}$$

The first set of constraints (x-variables) gives the flow balance for the active path; the second set of constraints (y-variables) gives the flow balance for the backup path. The variable $z_{u\upsilon}$ represents the amount of backup bandwidth reserved on link (u, υ). The first term in the objective function is the bandwidth consumed by the active path; the second term is the bandwidth consumed by the backup path. Note that the amount of backup bandwidth consumed on link (u, υ) is the largest value of $\theta_{ij}^{u\upsilon}$ for any link (i, j) on the active path. Note that if the integer program is infeasible, there is no feasible solution to the routing problem and the current request is dropped. We can introduce an additional constraint $x_{ij} + y_{ij} \le 1$ to explicitly take care of the fact that the active and backup paths are disjoint. This is current implicitly handled by setting $\theta_{ij}^{u\upsilon} = \infty$ if $(i, j) = (u, \upsilon)$: As outlined in [10], the quadratic constraint can be linearized.

Partial Information

This is the most practical case. The information available is the aggregate bandwidth used on each link by active paths denoted by F_{ij}, the aggregate bandwidth used on each link by backup paths denoted by G_{ij}, and the link residual bandwidths R_{ij}. Note that since we are only maintaining aggregate information on bandwidth usage, the amount of information being distributed to all nodes is independent of the number of LSPs that are currently using the network. Whereas the complete information scenario requires per-LSP information to be maintained, the partial information scenario requires information to be maintained only for two classes of LSPs: active and backup LSPs. This is only slightly more information than the no information model, which keeps track of only the total aggregate bandwidth usage. As we shall see later, using the developed routing algorithms, the small amount of extra information in the partial information scenario in comparison to the no information scenario can be used to obtain big gains in network performance fairly close to the complete information scenario when the performance metric is the number of rejected requests.

First note that some sharing of the backup paths is possible even though only minimal information is maintained. Let AP represent the active path and BP represent the backup path for the current demand. Let us assume for the moment that AP has been selected already. Let M represent the largest value of F_{ij} for some link (i, j) in the active path, that is,

$$M = \arg \max_{(i,j)\in AP} F_{ij}.$$

For a potential link (u, υ) on the backup path, if $M + b \le G_{u\upsilon}$, no additional bandwidth needs to be reserved on the backup path because any link failing on the active path generates a bandwidth need of at most $M + b$ on the links of the backup path. Recall that $G_{u\upsilon}$ is the amount of bandwidth in use by backups on link $u\upsilon$. If $M + b > G_{u\upsilon}$, since $G_{u\upsilon}$ units of bandwidth is shareable, only an additional reservation of $M + b - G_{u\upsilon}$ units is necessary. If this bandwidth is not available, this backup path is not feasible.

We can capture these sharing notions in a formulation similar to the complete information formulation. For partial information, we set the value of $\theta_{ij}^{u\upsilon}$ as follows:

$$\theta_{ij}^{u\upsilon} = \begin{cases} 0 & \text{if } F_{ij} + b \le G_{u\upsilon} \\ & \text{and } (i,j) \ne (u,\upsilon) \\ F_{ij} + b - G_{u\upsilon} & \text{if } F_{ij} + b > G_{u\upsilon} \\ & \text{and } R_{u\upsilon} \ge F_{ij} + b - G_{u\upsilon} \\ & \text{and } (i,j) \ne (u,\upsilon) \\ \infty & \text{otherwise.} \end{cases}$$

This is based on the observation that

$$\delta_{ij}^{u\upsilon} \le F_{ij} \quad \forall(i,j) \quad \forall(u,\upsilon).$$

One can solve the integer linear programming problem with the new values for $\theta_{ij}^{u\upsilon}$. However, a faster algorithm is needed for online routing in large networks. This is particularly so if the algorithm is to run on edge nodes with limited computational resources. Let us assume that *AP* has been determined already; hence, the value of M is known. In this case the cost $c_{u\upsilon}$ of using a link (u, υ) on the backup path is given by

$$c_{u\upsilon} = \begin{cases} 0 & \text{if } M + b \le G_{u\upsilon} \\ M + b - G_{u\upsilon} & \text{if } M + b > G_{u\upsilon} \\ & \text{and } R_{u\upsilon} \ge M + b - G_{u\upsilon} \\ \infty & \text{otherwise.} \end{cases}$$

One can solve a shortest path problem with the cost $c_{u\upsilon}$ on link (u, υ). This will result in the optimal backup path provided the active path has somehow been independently selected. Of course, the amount of sharing possible on the backup path influences the choice of active path, so the active path cannot be independently chosen at the outset.

Let $\overline{M} = \max_{(i,j)} F_{ij}$, represent the maximum active bandwidth that can be reserved on any link. Below, we give a high-level view of an algorithm solving the path selection problem under partial information. It assumes there is a subroutine DISJOINT PATH () that returns the optimal solution to the problem of finding two link disjoint paths between nodes s and d in the network where the cost of the first path is given by the sum of the a_{ij} for all links (i, j) on the first path, and the cost of the second path is given by the sum of c_{ij} for all links (i, j) on the second path. We define this problem more formally and give a solution procedure later in this section. Assuming that DISJOINT PATH () exists, the path selection algorithm is the following:

- STEP 1: Let $M = 0$.If $BEST = \infty$.
- STEP 2: If $M > \overline{M}$ then exit else compute the cost a_{ij} for using link (i, j) on the active path as

$$a_{ij} = \begin{cases} b & \text{if } F_{ij} \le M \\ \infty & \text{otherwise.} \end{cases}$$

Also compute c_{ij}, the cost to use link (i, j) on the backup path, as

$$c_{ij} = \begin{cases} 0 & \text{if } M + b \le G_{ij} \\ M + b - G_{ij} & \text{if } M + b > G_{ij} \\ & \text{and } R_{ij} \ge M + b - G_{ij} \\ \infty & \text{otherwise.} \end{cases}$$

- STEP 3: Solve DISJOINT PATH (). If the optimal solution to this problem is *OPT* and the value of $OPT < BEST$ then set $BEST = OPT$. Increment M and go to STEP 2.

Therefore, the feasibility of this approach to solving for the partial information case depends on the ability to solve DISJOINT PATH (). This problem can be stated more formally as follows: Given a directed graph and two costs a_{ij} and c_{ij} on link (i, j), find a pair of link disjoint paths between a given pair of nodes s and t with minimum total cost, where the cost of the first path will be the sum of the a_{ij} for all links (i, j) on the first path and the cost of the second path sum of the c_{ij} for all links (i, j) on the second path. Unlike the disjoint path problem considered in the no information scenario where the link costs are the same for both path computations, this problem with its different link costs a_{ij} and c_{ij} is NP-hard. The proof is in [11]. This article also gives an algorithm for this problem with a worst-case guarantee. However, the worst-case guarantee is too weak for our purposes. A more suitable dual-based algorithm is given in [12].

The basic idea in solving the overall problem is that once M is fixed, the problem becomes that of computing DISJOINT PATH (). Since we don't know the best choice of M, we try all values of M and take the least cost solution. The number of feasible M values is upper bounded by the number of network links, so trying all values of M is not very expensive.

This algorithm executes very fast even on large-sized networks and usually obtains solutions within 5 percent of the optimal solution. Several tricks can be used to speed up the algorithm even further. For example, instead of iterating from $M = 0$ to $M = \overline{M}$, it is enough only to do the computation for values of $M = f_{ij}$ for some link (i, j). Therefore, we have to call DISJOINT PATH () at most m times, where m is the number of links in the network. Of course, if different links have the same f_{ij} values, we need to do the experiment only once for all those links.

Node Failures

To extend the algorithm to work for node failures, we merely change the representation of nodes by splitting each node into an ingress subnode where all the incoming links terminate and an egress subnode where all the outgoing links terminate. The two subnodes are connected with a link, and the failure of this link is equivalent to a node failure.

Partial Information with Exact Reservations

Note that the use of aggregated (partial) information as described above causes the path computation algorithm to be conservative in its estimate of the amount of flow that will occur on a link in the backup path when a link in the active path fails. This is because the algorithm cannot determine which link in the active path will fail and hence has to assume the worst case, that is, the flow on a link in the backup path is taken to be the maximum of all the current active flows on links in the active path. This conservative value is used in computing the link costs for each link in the backup path (as explained above), and hence in determining the active and backup paths.

Once the paths are determined, LSP setup is done using signaling. We assume that the set of links in the active path is conveyed to every link in the backup path during the signaling phase. The motivation for doing this is the following: In the complete information model the values of $\theta_{ij}^{u\upsilon}$ depend only on knowing the set $\phi_{ij}^{u\upsilon} = A_{ij} \cap B_{u\upsilon}$. We do not need to know the sets A_{ij} and $B_{u\upsilon}$

To extend the algorithm to work for node failures, we merely change the representation of nodes by splitting each node into an ingress subnode where all the incoming links terminate and an egress subnode where all the outgoing links terminate.

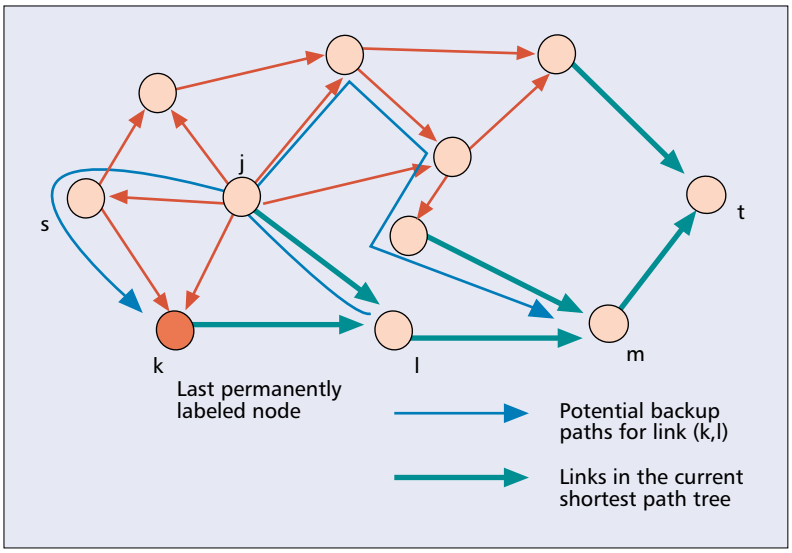

Figure 6. *Computing backup cost.*

individually. Therefore, when the links belonging to the active paths are passed along the backup path, any link (u, υ) on the backup path can update the set $\phi_{ij}^{u,\upsilon}$ for any link (i, j) in the active path. Note that this set changes only when link (i, j) is in the active path and link (u, υ) is on the backup path. This means that link (u, υ) on the backup path can compute the value of $\theta_{ij}^{u,\upsilon}$ (as in an earlier section), and can then make the exact reservation instead of the more conservative value used by the partial information model. The value of $\phi_{ij}^{u,\upsilon}$ is only known to the link (u, υ); hence, the path computation at the source is still done using partial information. The set $\phi_{ij}^{u,\upsilon}$ and hence the value of $\theta_{ij}^{u,\upsilon}$ changes when an LSP is removed from the network. Since there has to be signaling along both the active and backup paths to tear down the connection, each link on the backup path can update the value of $\theta_{ij}^{u,\upsilon}$ for all links (i, j) on the active path that is currently being torn down. Observe that in order to do all the computation above, a link (u, υ) only needs to know about the LSPs that use that link on the backup path. It does not need to have knowledge of any other LSP in the network. This lends itself well to the signaling protocols.

No Information with Exact Reservation

This is very similar to the setup described in the last section, except that partial information is not available to all the nodes in the network. In this case, the source node computes the shortest pair of disjoint paths, as in the no information case. One of these paths is designated the active path and the other the backup path. This information is signaled to the active and backup paths. In particular, the links in the active path are known to the backup path. Exact reservations are made by the backup path links as in the previous section.

Local Restoration

In this section we outline an algorithm for routing with local restoration. Instead of giving the details of the algorithm, we outline the key ideas involved in the design of the algorithm.

Backup Cost Computation using Shortest Path Iterations

First we consider the single link failure case, ignoring the intrademand sharing of backup bandwidth, under the partial and complete information models. Analyzing this case leads to an overall design of the algorithm. If link (i, j) is used in the active path, there has to be a backup path bypassing link (i, j) so as to back up any failure of this link. Note that the backup path has to start at node i but can terminate at any node between node j and the destination t on the active path. For now we ignore this, and consider the case where the backup path for link (i, j) has to start at i and terminate at node j. In this case, the overall bandwidth needed when link (i, j) is used in the active path is the sum of the bandwidth for using it in the active path and the bandwidth used for backing up the link. The bandwidth needed if link (i, j) is used on the active path is b. The bandwidth needed to back up link (i, j) can be computed as the shortest path from node i to node j after removing link (i, j).

We first consider robustness to single link failures. The cost of using link (u, υ) on the backup path if link (i, j) is used in the active path, denoted by $\theta_{ij}^{u,\upsilon}$, is the same as in the path restoration case, and depends on the information model used. After $\theta_{ij}^{u,\upsilon}$ is computed, we can determine the total cost of using link (i, j) in the active path. This total cost is the sum of the active path cost of link (i, j) and the cost of its bypass path. To determine the cost of bypassing link (i, j), we compute the shortest path from i to j (excluding link (i, j)) where the cost of each link (u, υ) in the path is given by $\theta_{ij}^{u,\upsilon}$. Let the length of this shortest path between i and j be ϕ_{ij}. Then the cost of using link (i, j) on the active path is $b + \phi_{ij}$, that is, the sum of the bandwidth usage on link (i, j) and bandwidth usage for bypass of link (i, j). Once usage costs are associated with each link in the network (using a total of m shortest path computations), we now compute the shortest path between s and t using $b + \phi_{ij}$ as the cost of link (i, j). This gives the minimum amount of bandwidth without intrademand sharing taken into account. Therefore, we totally solve $m + 1$ shortest path problems. This leads to the first design idea: *The cost of backup paths can be determined by solving shortest path problems, one for each link in the network.*

Reverse Shortest Path Computation

In the above discussion, we ignored the fact that the backup path for link (i, j) starts at i but can end at any node on the path from j to t (including j and t). Handling this case is facilitated by executing the shortest path algorithm backward from the destination to the source.

This facilitation is illustrated by Fig. 6, which shows a step in the backward execution of the algorithm. The dark lines in the graph represent the shortest path tree when Dijkstra's algorithm is executed backward from the destination. For every node that is permanently labeled in the shortest path tree there is a unique path from that node to the sink. Consider a node k that is permanently labeled when we are constructing the shortest path tree from the sink. Associated

with node k is the path $P(k) = \{l, m, t\}$ along the shortest path tree from node k to the destination. Consider link (k, j) in the network. The cost of using link (k, j) in the active path is the sum of bandwidth currently being routed and the cost of backing up link (k, j). The dotted lines in Fig. 6 illustrate three different paths to backup link (k, j). Previously, the cost of backing up link (k, j) was computed as the shortest path from k to j with $\theta_{kj}^{u\upsilon}$ as the cost of link (u, υ). Now instead of computing the shortest path from j to k we compute instead the shortest path from k to any node in $P(k)$. This can be done easily by running Dijkstra's algorithm from j using *quvjk* on link (u, υ), and terminating the algorithm when any node in the set $P(k)$ is permanently labeled by the algorithm. This example illustrates the reasoning leading to the second design principle: *It is necessary to execute the shortest path (Dijkstra's) algorithm backward starting at the sink.*

Intrademand Sharing Using State Information

To derive the next key idea, we consider the single link failure case where we also take into account intrademand sharing of backup bandwidth. Intrademand sharing of bandwidth occurs when link (i, j) uses link (u, υ) for a backup and reserves a bandwidth of w on link (u, υ). When some other link (k, l) on the active path wants to use link (u, υ) for its backup, in addition to any interdemand sharing it can use the already reserved bandwidth of w for free. Recall that the shortest path algorithm is to be run backward from the destination. In order to keep track of how much bandwidth is reserved at each link, we introduce an m-vector λ^u corresponding to node u in the network. (Recall that the number of links in the network is m.) λ_{ij}^u represents the amount of bandwidth reserved by the current demand for all the backup paths for all the links leading from node u to destination t. This bandwidth reservation for the current demand can be used to save bandwidth, by intrademand sharing, when backing up the links from u to the source s that are yet to be determined. Consider the case where the backup path for link (k, j) is being determined. Assume that the shortest path is being determined in the backward direction from node j to node k. The m-vector λ^j represents the reservation made for this demand for all the backup path from node j to the sink. This path is known since there is a unique path from node j to the sink in the shortest path tree. Consider a link (m, n) in the network. Define

$$\kappa_{mn} = F_{kj} + b - B_{mn} - \lambda_{mn}^j.$$

Then the cost of link (m, n) when determining the shortest backup path is given by

$$l_{mn} = \begin{cases} 0 & \text{if } \kappa_{mn} \le 0 \\ \delta_{mn} & \text{if } 0 \le \kappa_{mn} < b \text{ and } R_{mn} \ge \kappa_{mn} \text{ and } (m,n) \ne (k,j) \\ \infty & \text{otherwise.} \end{cases}$$

The cost in the case of the complete information case can also be modified similarly. Therefore, the above procedure gives us a method for accounting for the intra-demand sharing. This gives the third design principle: *Maintaining the m-vector at each node that gives us the amount of bandwidth reserved for the current demand for backing up all links from the given node to the destination can be used to account for intra-demand sharing.*

Adapting Algorithm for Node Failures

To get node bypass paths, the procedure is almost the same as the edge bypass path case. There are two main differences. The first is that when we want to determine the cost of including link (i, j) in the active path, we have to determine the cost of finding a backup from node i to the successor of node j without using any of the links incident on node j. Of course, when the algorithm is run backward from the sink, the successors of all the nodes that are permanently labeled by Dijkstra's algorithm are already known since a path has been established from that node to the destination. The second important difference is that when a node fails, all the links incident on the node fail. Therefore, the cost of the backup has to account for all the links failing simultaneously. We only consider the outgoing links from the node. For example, when computing the cost of using link (i, j) in the active path, we have to consider the cost of backing up demands that use link (j, l) for $l \in V$. Therefore, the cost of all links (without considering intrademand savings) have to be modified as follows:

$$\theta_{ij}^{u\upsilon} = \begin{cases} 0 & \text{if } \sum_{(j,k)\in E} \delta_{ij}^{u\upsilon} + b \le G_{u\upsilon} \text{ and } (i,j) \ne (u,\upsilon) \\ \sum_{(j,k)\in E} \delta_{ij}^{u\upsilon} + b - G_{u\upsilon} & \text{if } \sum_{(j,k)\in E} \delta_{ij}^{u\upsilon} + b > G_{u\upsilon},\ R_{u\upsilon} \ge \sum_{(j,k)\in E} \delta_{ij}^{u\upsilon} + b - G_{u\upsilon} \text{ and } (i,j) \ne (u,\upsilon) \\ \infty & \text{otherwise.} \end{cases}$$

Cost of using link (u, υ) for partial information case:

$$\theta_{ij}^{u\upsilon} = \begin{cases} 0 & \text{if } \sum_{(j,k)\in E} F_{jk} + b \le G_{u\upsilon} \text{ and } (i,j) \ne (u,\upsilon) \\ \sum_{(j,k)\in E} F_{jk} + b - G_{u\upsilon} & \text{if } \sum_{(j,k)\in E} F_{jk} + b > G_{u\upsilon} \text{ and } R_{u\upsilon} \ge F_{ij} + b - G_{u\upsilon} \text{ and } (i,j) \ne (u,\upsilon) \\ \infty & \text{otherwise.} \end{cases}$$

The next algorithm design idea is then the following: *Modifying the cost of the links in the computation of the backup costs can be used to account for node failures*. The description of the complete algorithm using these ideas is in [12].

Illustrative Experimental Result

This section gives an example experimental result illustrating the sharing efficiency and performance obtained in the different information

Maintaining the m-vector at each node that gives us the amount of bandwidth reserved for the current demand for backing up all links from the given node to the destination can be used to account for intra-demand sharing.

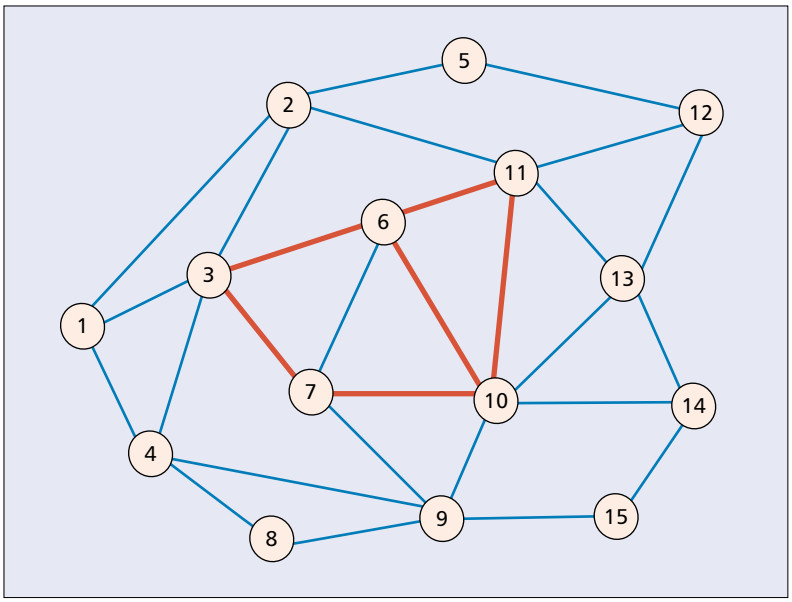

Figure 7. *The 15-node test network.*

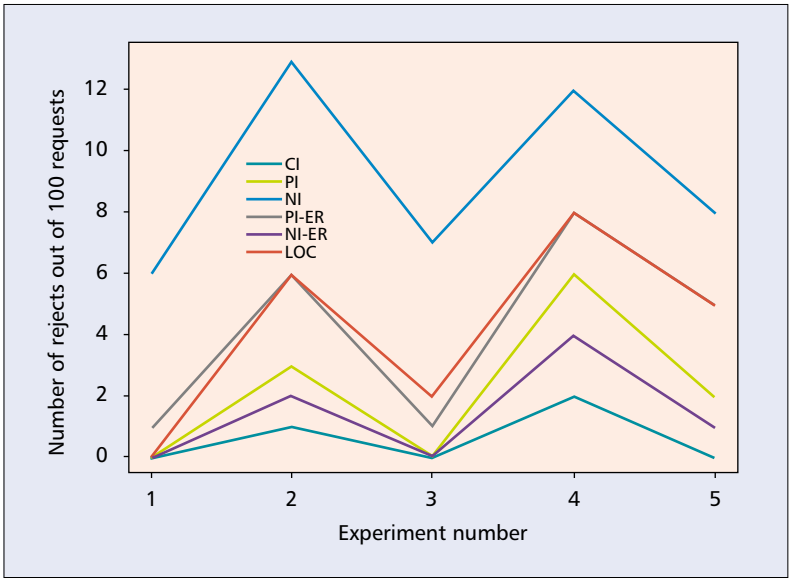

Figure 8. *Number of rejected requests for 5 random experiments in the 15-node test network.*

scenarios. We show that making exact reservations leads to very good performance, making the case of partial information with exact reservations both practical and efficient. The experimental setup is as follows. We performed experiments on a network with 15 nodes and 56 links. The 15-node network is shown in Fig. 7. Each undirected link in the figure represents two directed links. The capacity of each of the thin links is 60 units and of each thick link is 240 units. In each experiment LSPs with bandwidth requests uniformly distributed in the range [6, 9] arrive into the network. The source and destination of the LSPs are chosen at random. The objective of the experiments was to determine the number of rejected requests out of the 100 requests that arrive in the network. We performed five experiments. There were six algorithms run on the same data:

- Complete Information (CI)
- Partial Information (PI)
- No Information (NI)
- Partial Information-Exact Reservation (PI-ER)
- No Information-Exact Reservation (NI-ER)
- Local Restoration to protect against single link failures (LOC)

The results are shown in Fig. 8. Note that exact reservation modes perform much better than the case where reservations are not exact. Furthermore, among the path restoration algorithms, Partial Information with Exact Reservation performs the best and is fairly close to complete information. Partial information (with nonexact reservation) performed better than no information with exact reservation. This perhaps is a function of the topology. We expect NI-ER to be reasonably good but inferior to PI-ER. The performance of LOC also looks very competitive.

Signaling and Routing Protocol Extensions

As discussed before, the information in the PI model is only an incremental addition to the NI model. In the NI model a link state routing protocol with traffic engineering extensions is used to indicate the bandwidth on each link that has been reserved by (active) paths already set up. The only additional information needed for the PI model is to include the bandwidth reserved on each link by backup paths that have already been set up. This can easily be accomplished by modifying the OSPF opaque LSA used to convey traffic engineering capabilities so that it now has a subtype indicating the link bandwidth allocated for backup.

In MPLS networks, a protocol like RSVP-TE or CR-LDP is used for LSP setup. In addition to the capabilities already incorporated in a protocol like RSVP-TE, the main additional information needed is that the active path information (conveyed by the explicit route object, ERO) be sent during backup path setup to links on the backup path. This permits links on the backup path to make exact reservations for backup when using only the PI model.

Concluding Remarks

QoS routing research has been mostly focused on the routing of a single path without any fault tolerance requirements. Due to potential applications in MPLS networks, routing of QoS guaranteed restorable paths has become an issue of recent research interest. This article discussed some recently developed algorithms for restorable routing of bandwidth guaranteed paths. Also, the possible information models for link usage, and incremental additions to signaling and routing protocols were discussed. A method for routing with partial or aggregate information but with exact reservations of shared backup bandwidth was proposed as being both feasible and efficient for the routing of restorable bandwidth guaranteed paths. This scheme can be extended to handle the case of failures associated with shared risk link groups.

REFERENCES

[1] R. Guerin, D. Williams, A. Orda, "QoS Routing Mechanisms and OSPF Extensions," *Proc. GLOBECOM 1997*.

[2] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with Applications to MPLS Traffic Engineering," *IEEE JSAC*, Special Issue on Quality of Service in the Internet, Dec. 2000.

[3] S. Plotkin, "Competitive Routing of Virtual Circuits in ATM Networks," *IEEE JSAC*, 1995, Special Issue on Advances in the Fundamentals of Networking, pp. 1128–36.

[4] I. Matta and A. Bestavros, "A Load Profiling Approach to Routing Guaranteed Bandwidth Flows," *Proc. IEEE INFOCOM*, Apr. 1998.

[5] B. Davie and Y. Rekhter, *MPLS Technology and Applications*, Morgan Kaufman, 2000.

[6] V. Sharma *et al.*, "Framework for MPLS based Recovery," Internet draft, draft-ietf-mpls-recovery-frmwrk-04.txt, July 2001.

[7] J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, vol. 14, 1984, pp. 325–36.

[8] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum Interference Routing for Restorable Bandwidth Guaranteed Connections," *Proc. INFOCOM '02*, June 2002.

[9] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," *Proc. INFOCOM 2000*, Apr. 2000.

[10] H. D. Sherali and W. P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Non-convex Problems*, Kluwer, 1999.

[11] C. Li, S. T. McCormick, and D. Simchi-Levi, "Finding Disjoint Paths with Different Path Costs: Complexity and Algorithms," *Net.*, vol. 22., 1992, pp. 653–67.

[12] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information," *Proc. INFOCOM 2001*, Apr. 2001.

BIOGRAPHIES

MURALI KODIALAM [M] (muralik@bell-labs.com) obtained a Ph.D. in operations research from Massachusetts Institute of Technology in 1991. He has worked at Bell Laboratories since October 1991. He is currently in the High Speed Networks Research Department, working on resource allocation and performance of communication systems including routing in MPLS systems, topology construction and routing in ad hoc wireless networks, and reliable routing in optical networks. He is a member of INFORMS.

T. V. LAKSHMAN (lakshman@research.bell-labs.com) received his Ph.D degree in computer science from the University of Maryland, College Park. Prior to that he received a Master's degree from the Department of Physics, Indian Institute of Science, Bangalore. He is currently a director in the Networking Research Laboratory at Bell Laboratories. Previously, he was at Bellcore (now Telcordia) where he was most recently a senior research scientist and technical project manager in the Information Networking Research Laboratory. His recent research has been in issues related to traffic characterization and provision of quality of service, architectures and algorithms for gigabit IP routers, end-to-end flow control in high-speed networks, traffic shaping and policing, switch scheduling, and routing in MPLS and optical networks. He is a co-recipient of the 1995 ACM Sigmetrics/Performance Conference Outstanding Paper Award, and the IEEE Communications Society 1999 Fred. W. Ellersick Prize Paper Award.

QoS routing research has been mostly focused on the routing of a single path without any fault-tolerance requirements. Due to potential applications in MPLS networks, routing of QoS guaranteed restorable paths has become an issue of recent research interest.