A Novel Two-Step Approach to Restorable Dynamic QoS Routing

Ji Li, *Student Member, IEEE*, and Kwan Lawrence Yeung, *Senior Member, IEEE*

***Abstract*—Aiming at minimizing the combined bandwidth cost of a pair of disjoint active and backup paths, a popular approach to designing restorable dynamic quality of service (QoS) routing schemes is based on the integer linear programming (ILP) formulation. Owing to the very different natures of active and backup paths, we found this approach problematic. In this paper, we propose an alternative approach, called two-step restorable QoS routing. In the first step, an active path is found using the widest shortest path (WSP) routing. In the second step, the corresponding backup path is determined using one of the three variants of shortest widest path (SWP) routing: basic SWP, approximate SWP or composite SWP. Combining the two steps, three novel two-step routing algorithms, denoted by SBW, SAW, and SCW, are obtained. Comparing with the best known algorithms, we show that our two-step routing approach yields noticeably lower call blocking probability, shorter active-path length, and additional flexibility of adjusting backup-path length (depending on the SWP variant adopted). Besides, our two-step routing approach gives a much shorter running time than the ILP approach, which makes it more suitable for dynamic routing.**

***Index Terms*—MPLS, restorable dynamic routing, two-step routing.**

I. Introduction

THE MIGRATION of mission-critical traffic, such as business data, to Internet Protocol (IP) networks means that modern IP networks must provide reliable transmission. Multiprotocol label switching (MPLS) [1] is a technology of choice in future IP-based transport networks. To deliver reliable services, networks such as MPLS require efficient recovery schemes [2] to provide protection of the traffic carried on different data paths. The basic idea is that for each pair of communicating endpoints, backup paths are provisioned to protect the traffic carried on the active path. If the active path fails, transmission can be restored by redirecting the protected traffic to the backup paths. Under the assumption that there can be only one failure that happens or exists at any given time, typical recovery schemes are designed to fully recover from any single network fault.

A. Existing Recovery Schemes

Building a fault-resilient network is not simple. Various recovery mechanisms have been proposed [3]–[5], [10]–[12]. They differ from each other in the speed of recovery, the amount of resources that must be preallocated (if any) to backup paths, the increased complexity of configuration and signaling, and the change in the length of data paths.

Depending on how backup paths are established, recovery schemes can be grouped into two categories: rerouting or protection switching. Using rerouting, backup paths are created on demand after detecting a fault on the active path. The associated recovery time is, in general, long due to the signaling and processing overhead in establishing backup paths on demand. With protection switching, backup paths are preestablished. A rapid recovery, which is crucial for mission-critical traffic, is possible and the packet loss during the switchover can be minimized.

Another way to categorize recovery schemes depends on where the recovery takes place, local or end-to-end. Local repair is to protect against a link or neighbor node fault, and to minimize the amount of time required for failure propagation. In local repair, the node immediately upstream of the fault detects the fault and also initiates the recovery. This allows almost zero fault notification time. The associated recovery follows the rerouting model if the backup path is established on demand, or protection switching model if the backup path is preestablished.

End-to-end recovery is also known as path protection, where a single backup path (which can be preestablished or established on demand) is used to protect against any link or node fault of an active path. To achieve this, the backup path must be disjoint with the active path. Unlike local repair, a nonnegligible amount of fault notification time is required.

B. Restorable Dynamic QoS Routing

In this paper, we focus on on-demand path protection. Assume that no prior knowledge about future call requests is available. Each new call request arrives with a predetermined QoS/bandwidth requirement. An on-demand path computation, also known as restorable dynamic QoS routing algorithm [10], is then triggered for finding a pair of disjoint active and backup paths to carry the call. If the network does not have enough resources to simultaneously carry both paths, the call will be blocked. We assume that no ongoing calls can be rerouted to make room for the new call, as this involves significant amount of signaling overhead, and may also adversely affect the QoS of the ongoing calls.

Since the resources of backup path can be shared, if more network-state information is known, it is more favorable to

Manuscript received July 13, 2004; revised May 31, 2005. This work was supported by Hong Kong Research Grant Council Earmarked Grant HKU 7150/04E.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: jili@eee.hku.hk; kyeung@eee.hku.hk).

Digital Object Identifier 10.1109/JLT.2005.856227

bandwidth sharing. There exist three general network-state information models.

1) No sharing (NS) [4]: Only the residual bandwidth of each link is known by every node.
2) Partial information (PI): In addition to the residual bandwidth, every node has some partial knowledge about the existing calls. In particular, sharing with partial routing information (SPI) [4], [7] assumes that the knowledge of the total amount of bandwidth assigned to the aggregated active paths or backup paths on each link is known, and distributed partial information management (DPIM) [12] assumes each node maintains some extra estimation information for remote links.
3) Complete/full information: Complete knowledge of all network state and activity is known. There are a few proposals fall into this category. They all strive to make the complete-information model practical by minimizing the overhead involved in collecting information. Specifically, the complete per-flow information is maintained by every node following the sharing with complete routing information (SCI) [4], [7] scheme. Successive survivable routing (SSR) [10] can obtain the similar information as SCI, but is much simpler by using a centralized matrix-based information-aggregation scheme. On the other hand, full information restoration (FIR) [11] adopts a distributed method to collect the necessary network information to achieve accurate sharing.

In this paper, we focus on the complete-information model. It allows the best network performance, and the information-collection process can be made very efficient by following the approaches used in SSR or FIR. Besides, the algorithms we proposed in this paper can be easily extended to other information models.

Under the complete-information model, an efficient restorable routing algorithm was designed in [4] and [7] based on integer linear programming (ILP) formulation, aiming at minimizing the sum of bandwidth cost (defined in Section III) consumed by the pair of disjoint active and backup paths. We call it Kodialam's algorithm. Recently Xiong *et al.* [5] enhanced Kodialam's algorithm by using a more efficient ILP formulation. Xiong's algorithm differs from Kodialam's in two major ways: 1) Xiong's objective function is to minimize a weighted sum of bandwidth cost consumed by the pair of active and backup paths, in which the backup-path bandwidth cost is weighted less; and 2) a nonzero virtual bandwidth cost is introduced to a link (for carrying a backup path), which would otherwise have a zero cost in Kodialam's algorithm. Simulation results [5] showed that the weighting factor can help to save bandwidth and the virtual cost can help to reduce the backup-path length.

Another approach of designing restorable QoS routing adopts two separated routing stages, such as FIR routing algorithm [11]. For each call request, FIR first computes the shortest path between the source and destination nodes as their active path. It then assigns each link (except those carry the active path) a cost, which is similarly defined as in Kodialam's algorithm. The backup path is given by the corresponding minimum-cost path.

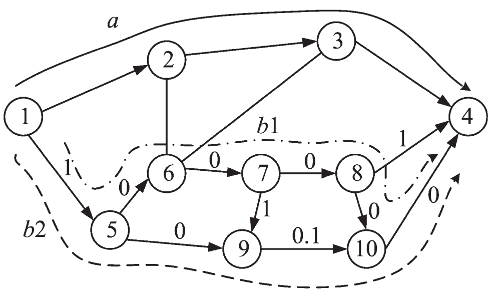

Fig. 1. Active path a has two potential backup paths $b1$ and $b2$. The number above each link denotes the (residual bandwidth) cost of setting up the backup path on that link.

C. Our Contributions

Instead of relying on the computationally expensive ILP formulation, we follow a simple two-step approach in designing restorable dynamic routing algorithms in this paper. The first step is optimized for finding the active path using the widest shortest path (WSP) routing algorithm. Once the active path is found, the second step is to determine the backup path using some variants of the shortest widest path (SWP) routing algorithm. In this paper, three variants of SWP are proposed, namely, basic SWP, approximate SWP, and composite SWP. Combining both steps, three novel two-step routing algorithms, denoted by SBW, SAW, and SCW, are designed. Comparing with the best known existing algorithms, we show that the two-step routing algorithms yield noticeably lower call blocking probability, shorter active-path length, and additional flexibility of adjusting backup-path length (depending on the SWP variant adopted). Last but not the least, the running time of our two-step algorithms has a definite edge over the ILP approach, which makes it more suitable for dynamic routing.

The rest of the paper is organized as follows. Section II highlights some potential problems of the existing approaches to restorable dynamic routing. Section III presents the major assumptions, notations, and definitions to be used throughout the paper. Section IV describes the three proposed two-step restorable dynamic routing algorithms in details. Their performance is compared with Xiong's, Kodialam's, and FIR algorithms in Section V. Finally, we conclude the paper in Section VI.

II. Some Observations on Existing Algorithms

Kodialam's algorithm greedily chooses those links with zero residual bandwidth cost for backup path independent of its length. This produces long backup paths, which can increase the call blocking probability. We illustrate this using the example shown in Fig. 1, which is adopted from [5]. Assume there are two call requests from node 1 to node 4, and only path 1–2–3–4 has enough resources to carry the active paths for both calls. Further assume that only the link 1–5 has the ability to carry the backup paths for both calls while the other links (of course except those on path 1–2–3–4) can carry just one. Let path 1–2–3–4 be selected as the active path for the first call, denoted as path a. Path a has two candidate backup paths, $b1$

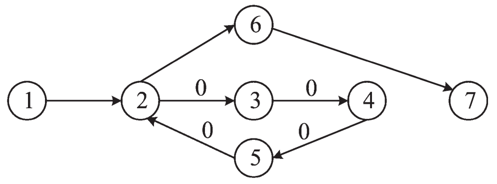

Fig. 2. Backup path 1–2–3–4–5–2–6–7 (with loop) has the same cost as backup path 1–2–6–7 (without loop). Due to zero link cost, the flow balance constraints [4] in Kodialam's algorithm cannot prevent the ILP from generating the former path as the solution.

and $b2$. $b1$ is longer but costs less. With Kodialam's algorithm, $b1$ will be chosen. When the second call arrives, its active path can again be established along path 1–2–3–4, but the call will be eventually blocked as a result of no feasible backup path being found. However, if $b2$ (which is shorter but costs more than $b1$) is selected as the backup path for the first call, then the second call can be successfully established with a backup path along 1–5–6–7–8–4.

Since the cost of links along backup paths can be zero, the flow balance constraints [4] in Kodialam's algorithm, for ensuring that the number of incoming flows of each node is equal to the number of outgoing flows, cannot prevent all backup-path loops. This can be seen from the example in Fig. 2. We need to point out that loops in Kodialam's algorithm can be removed by adding an additional constraint that every node cannot appear more than once in any active or backup path. For a fairer comparison, this constraint is also added in our implementation of Kodialam's algorithm in Section V.

Note that Xiong's algorithm [5] does not have this looping problem because a virtual cost is assigned to replace the zero link cost. In [5], virtual cost[1] is the product of an arbitrary number μ and bandwidth requirement w of each incoming call. In order to reduce the length of backup paths, it is suggested that μ should be in the range of (0, 1]. In practice, setting an appropriate value to parameter μ is difficult. Sometimes, this may give unexpected results. Let us consider the example in Fig. 3. Assume there is a call request from source node 1 to destination node 8 with bandwidth requirement $w = 10$ (units), and parameter μ is set to 0.4. (These values are borrowed from the simulation settings in [5].) Fig. 3 shows two candidate backup paths $b1$ and $b2$ for the active path (not shown). Obviously, $b1$ should be selected as it has lower cost (20) and shorter length (four hops). However, Xiong's algorithm assigns virtual cost $0.4 \times 10 = 4$ to the links connecting nodes 2 and 3, and nodes 3 and 7. As a result, the less optimal path $b2$ will be chosen.

FIR algorithm [11] first computes the active path with shortest path algorithm, and then assigns each link a weight (like Kodialam's algorithm). The backup path is then given by the corresponding minimum-cost path. Similar to Xiong's algorithm, FIR also gives a virtual cost for zero link cost. This virtual cost is a predefined small number so that among multiple paths with zero cost, the minimum hop path will be chosen.

[1]According to the authors' e-mail correspondence with us, the virtual cost is the product of an arbitrary number μ and the unit of required bandwidth, which is inconsistent with the expressions in [5]. Nevertheless, the virtual cost in [5] could work better for its purpose to reduce the length of backup path in the general case.

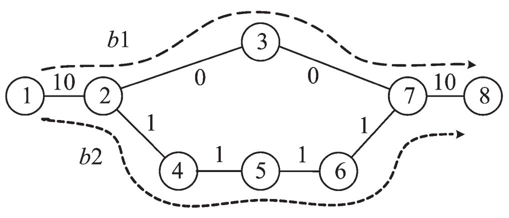

Fig. 3. Arbitrary parameters in Xiong's algorithm may cause an inappropriate path to be chosen. The number above each link denotes the (residual bandwidth) cost of setting up the backup path on that link.

However, FIR still faces the problem of picking a very long but "cheapest" backup path if the virtual cost is unsuitably chosen, and the most suitable value of virtual cost is most likely to change with the network loads. Therefore, FIR also has difficulty in setting the virtual cost, just like Xiong's algorithm.

III. Assumptions, Notations and Important Definitions

In this section, we summarize the major assumptions, notations, and definitions to be used throughout the paper. We follow the common practice [4], [5] of only requiring that the backup path be link disjoint with the active path. This can be explained by the facts that colocated failover node devices can be easily installed for node protection. Nevertheless, we can easily extend our proposed routing algorithms (in Section IV) to node disjoint scenario.

Assume each path setup request arrives with a predetermined bandwidth requirement w. For each link l, its physical bandwidth consists of three parts: A_l, B_l, and R_l. A_l is the total amount of reserved bandwidth dedicated to all active paths carried by link l. Such resources cannot be shared. B_l is the total bandwidth occupied by all backup paths on link l. B_l can be shared by some backup paths, provided that their associated active paths are disjoint. For example, if two active paths share a common link (so they are not disjoint), both will be affected by a single network fault occurring on this common link. Given that their backup paths also share a common link, the amount of backup bandwidth they reserved on the common link must not be shared. Otherwise, the backup paths cannot provide sufficient bandwidth to carry the protected traffic from both flows. Finally, the residual bandwidth R_l is the difference between the physical bandwidth on link l and the total consumed bandwidth $(A_l + B_l)$.

We can see that for any future active path set up on link l, R_l is the only available bandwidth that can be used. If $R_l \geq w$, the active path can be set up with a link cost equal to w. The total cost of setting an active path, or its path cost, is the sum of the costs induced at individual links along the selected path. We can see that the path cost is minimized if the hop distance between the source and the destination is minimized.

For setting up a backup path on link l for a new active path a, the available bandwidth $S_l(a)$ consists of two components, residual bandwidth R_l, and the portion of B_l that can be shared to carry this backup path, denoted by $\gamma_l(a)$. That is

$$S_l(a) = \gamma_l(a) + R_l. \tag{1}$$

If $S_l(a) \geq w$, the backup path can be set up. To encourage backup-bandwidth resources sharing, the associated link cost is

$$\text{link cost} = \begin{cases} 0, & \text{if } \gamma_l(a) \geq w \\ w - \gamma_l(a), & \text{if } 0 < \gamma_l(a) < w \\ w, & \text{if } \gamma_l(a) = 0 \end{cases}. \tag{2}$$

Note that the link cost is infinite for those links traversed by active path a or with $S_l(a) < w$. It should be noted that $\gamma_l(a)$ is always consumed with higher priority than B_l. In fact, B_l is used only if $\gamma_l(a)$ is not enough. Unlike active paths, the path cost of a longer backup path may cost less than that of a shorter one, because of bandwidth sharing.

Next, we derive $\gamma_l(a)$, the portion of backup bandwidth B_l that is subject to sharing. Given that we have two links m and l, let $\mathcal{A}^m$ be the set of active paths carried on m. Let $\mathcal{A}_l^m$ be a subset of $\mathcal{A}^m$ that has its backup paths passing through link l. So, from link l's point of view, active paths in $\mathcal{A}_l^m$ are not disjoint and thus, they cannot share their reserved backup-path resources on link l. Let the total amount of backup bandwidth reserved for all active paths belonging to $\mathcal{A}_l^m$ on link l be ξ_l^m. Then, $\gamma_l(a)$ is given by

$$\gamma_l(a) = B_l - \max_{m \in a} \xi_l^m. \tag{3}$$

The second term $\max_{m \in a} \xi_l^m$ on the right-hand side is to take the maximum of ξ_l^m over all possible links (m) along the active path a, which are protected by the backup path on link l. Since $\max_{m \in a} \xi_l^m \leq B_l$, we have $\gamma_l(a) \geq 0$.

If enough bandwidth resources can be reserved to carry the pair of active and backup paths, the current call request is accepted. Otherwise, the request is blocked.

IV. Two-Step Routing

A. Motivations

We found that the existing algorithms can be improved if the following issues can be properly addressed.

1) The resource reserved for an active path is dedicated and that of a backup path is shared. A backup path will not be used unless there is a switchover from some protected active path. Simply minimizing the sum of the resources reserved for both active and backup paths, as in Kodialam's algorithm, is not fair, as the contribution from the active path should certainly deserve a heavier weight. Although a weighted sum is used in Xiong's algorithm, it is difficult (if not impossible) to determine the optimal value for the weighting factor. In [5], the naïve trial-and-error approach is adopted.
2) In the ILP approach, due to the simultaneous optimization of the total bandwidth cost consumed by both active and backup paths, the dedicated resource consumed by active paths alone is not minimized.
3) In Kodialam's algorithm, a backup path with zero cost will be chosen independent of its length. This results in long backup paths. To address this problem, a nonzero cost, in both Xiong's and FIR algorithms, is introduced to a link that would otherwise have a zero cost in Kodialam's algorithm. However, the way of determining the nonzero cost is quite *ad hoc* (as shown in Section II).
4) The resource reserved for an active path is released as soon as the carried call is finished. The resource reserved for a (shared) backup path is released only when all active paths that are protected by this backup path are released. The impact of different resource holding times should be properly reflected in the designed objective function.
5) There is no effort in load balancing the carried traffic in the network. Load balancing can help to maximize the network potential in admitting future calls, thus minimizing call blocking probability. This is because wider links are less likely to get saturated.

As an effort to properly address the above issues, we believe that two dedicated routing algorithms operating in series should be adopted, one optimized for active paths and the other optimized for backup paths. We call this approach two-step restorable QoS routing.

Since the active path is dedicated to a particular call and is occupied for the whole call duration, the resource consumed by an active path is significant and should be minimized whenever possible. Following this argument, a WSP algorithm is proposed for routing active paths in the first step, as detailed in Section IV-B.

On the other hand, the backup path is shared and will not be occupied unless there is a fault in the protected active path; it is considered more important to facilitate load balancing rather than minimizing the reserved bandwidth. SWP routing is preferred as it allows us to more evenly distribute the backup paths/loads over the whole network. In this paper, three variants of SWP algorithms are proposed for determining the backup paths in the second step. They are

1) basic SWP algorithm (detailed in Section IV-C);
2) approximate SWP algorithm (detailed in Section IV-D); and
3) composite SWP algorithm (detailed in Section IV-E).

B. WSP Routing for Active Paths

With bandwidth saving as the primary objective and load balancing the second, the WSP routing algorithm is chosen for active paths. If multiple shortest active paths between a source–destination pair are found, the widest path is chosen. Since R_l is the only available bandwidth for carrying active paths, the width of an active path is determined by the minimum R_l of all links on the path, i.e., its bottleneck bandwidth; and the widest path is the path that has the largest bottleneck on link residual bandwidth among all available paths from the source to the destination. In so doing, "shortest path" ensures that the resource consumed by the active path (and so is the end-to-end delay) is minimized (first), and "WSP" helps to balance the load in the network (subsequently).

In practice, when a call request with bandwidth requirement w arrives, we first remove all links whose R_l is less than w to produce an abridged topology. Then, we apply the WSP algorithm to find the active path. In [9], an implementation of

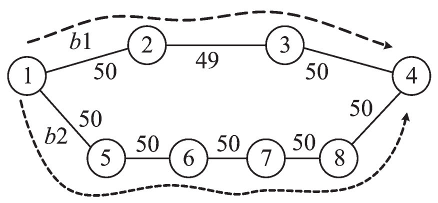

Fig. 4. Approximate SWP can yield a better performance. The number above each link is the total available bandwidth $S_l(a)$ of that link, not its link cost.

WSP based on Bellman–Ford algorithm was proposed. Alternatively, WSP can be implemented by modifying the Dijkstra's algorithm to find the shortest path with the widest bottleneck link R_l based on two metrics [6]: hop count and residual bandwidth. So, in the modified Dijkstra's algorithm, we always mark the next node, which has the minimal hop count and with the largest maximal residual bandwidth.

C. Basic SWP Routing for Backup Paths

For routing backup paths, we consider load balancing as the primary objective and minimizing backup-path cost the second. SWP routing [8] is chosen. With SWP, if there are multiple widest paths that are disjoint with the active path obtained using WSP in the first step, the path with the shortest distance is chosen. It should be emphasized that the available bandwidth for carrying a backup path on each link is obtained from (1) and (3). As a result, finding the set of "widest paths" first ensures load balancing, and finding the "SWP" next minimizes the reserved resources. In order to distinguish with the variants of the SWP algorithms to be proposed later, we call the vanilla SWP adopted here as basic SWP.

Our implementation of the basic SWP follows that in [6]. First, we prune the links for insufficient resources, i.e., links with $S_l(a) < w$ (note that the resources taken up by the active path found in step 1 should have already been considered). Then, we apply Dijkstra's algorithm to find a widest path based on $S_l(a)$ defined in (1). Let *T* be the minimum of all $S_l(a)$ along the widest path found. We cut off the links with $S_l(a) < T$ to generate another pruned topology. Finally, we apply Dijikstra's algorithm to find the shortest hop-distance path.

D. Approximate SWP Routing for Backup Paths

An SWP may waste much bandwidth when it strictly picks up links higher than the required threshold *T*. Consider the example in Fig. 4 for setting up a backup path with $w = 3$ between nodes 1 and 4. The number above each link is the total available bandwidth $S_l(a)$ of that link (not its link cost). For simplicity, assume $\gamma_l(a) = 0$ for all the links such that no sharing of backup-path bandwidth is possible.

If basic SWP routing is used, *T* will be set to 50. As a result, the link connecting nodes 2 and 3, which has a bandwidth less than 50, will be pruned. The final backup path produced by basic SWP is $b2$, with a total path cost of $3 \times 5 = 15$. If we can lower the value of *T* to 49, the same basic SWP will return the backup path $b1$, which has a much shorter length, three hops, and a much lower cost: 9.

In short, a slight relaxing on the "widest" path selection criterion (T) can give a significant gain in backup-path cost and length. Here, we propose an enhanced algorithm, called approximate SWP, to capture this effect. In approximate SWP, a ratio δ $(0 \leq \delta \leq 1)$ is defined to lower the *T* threshold (by a factor of δ) to enlarge the set of candidate backup paths. When $\delta = 0$, approximate SWP degrades into the simple shortest path algorithm. When $\delta = 1$, approximate SWP is the same as the basic SWP in Section IV-C.

E. Composite SWP Routing for Backup Paths

In both basic SWP and approximate SWP routing algorithms, the widest path is selected based on the total available bandwidth for a backup path $S_l(a)$. From (1), we can see that $S_l(a)$ consists of two components, equally weighted. If two links/paths have the same value of $S_l(a)$ but different values of each component, basic SWP and approximate SWP will treat both links/paths as equally desirable.

Since the residual bandwidth component R_l can be used to admit both active and backup paths, it is more precious and should therefore be consumed with lower priority than the other component $\gamma_l(a)$, which is the portion of backup bandwidth subject to sharing. Another variant of the basic SWP is thus designed. We call it composite SWP routing algorithm. It differs from the earlier basic and approximate SWP only in the procedure of finding the shortest path. In basic or approximate SWP, a pruned network is obtained by first removing links traversed by the active path a or with $S_l(a) < w$, and then links whose $S_l(a)$ is less than *T* (or $\delta^* T$). The distance between two nodes in the resulting pruned topology is measured by their hop distances. In composite SWP, the distance between two nodes is redefined. In particular, each link is now associated with a heuristic distance D_l, where

$$D_l = \frac{R_l}{1 + \gamma_l(a)}. \tag{4}$$

The idea is to assign the link with a larger component value of $\gamma_l(a)$ with a shorter distance; thus, it will be included in the selected shortest path with a higher probability. We can see that the composite SWP algorithm can save the residual bandwidth, but at a cost of a slight increase in the backup-path length (measured in hops).

F. Three Two-Step Routing Algorithms

Now, we can combine step 1 WSP routing for active paths, with the three variants of the SWP routing for backup paths, to form three two-step restorable dynamic QoS routing algorithms. Note that in step 1, the primary function is to find the shortest path; whereas in step 2, the primary function is to find a widest path. For simplicity, we denote each combined two-step routing algorithm by a three-letter acronym following the convention of SxW, where x denotes the version of the WSP being used in step 2. So we have SBW, SAW, and SCW algorithms, corresponding to the case that basic, approximate, or composite SWP is used.

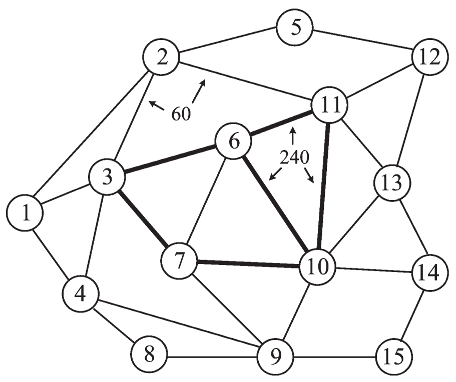

Fig. 5. Simulation Topology I—Kodialam's 15-node test network.

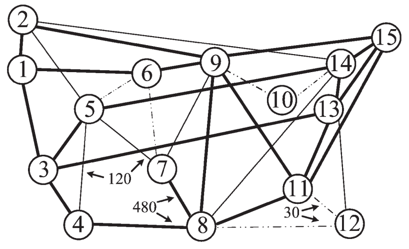

Fig. 6. Simulation Topology II—U.S. Sprint backbone network.

V. Simulation Results

In this section, we compare the performance of our proposed two-step routing algorithms with Kodialam's, Xiong's, and FIR algorithms. In particular, the following performance measures are used: call blocking probability, active-path length, and backup-path length. Among the three, call blocking probability is the most important measure, as it directly reflects the traffic-carrying capability of a network. Active-path length ranks next, as it determines the end-to-end delay performance experienced by the user traffic. Backup-path length is probably not as important as the previous two because it only affects the performance of the user traffic when a network fault occurs.

Two simulations are conducted based on two different network topologies, as shown in Figs. 5 and 6.

1) Topology I (Fig. 5) is adopted from [4], [5], and [7], consisting of 15 nodes and 28 bidirectional links. Two types of links are in Topology I: heavy links of 240 units of bandwidth (in each direction), and light links of 60 units. The bandwidth requirement of each call w is uniformly distributed between three and eight units for Topology I.
2) Topology II (Fig. 6) is based on the U.S. Sprint backbone network. It has 15 nodes and 33 bidirectional links. Three types of links exist: heavy links of 480 units of bandwidth (modeling OC-48), light links of 120 units (modeling OC-12), and dashed links of 30 units (modeling OC-3). In Topology II, we assume the bandwidth requirement w is uniformly distributed between three and six units.

We assume that calls arrive one by one and call holding time is assumed long enough, so we can consider accepted calls do not leave[2] [4], [5], [7]. The source and destination nodes of every call are picked up randomly. For the SAW algorithm, we set parameter δ of the approximate SWP to 0.95. For Xiong's algorithm, we set its parameters $\varepsilon = 0.1$ and $\mu = 0.2$, as recommended in their paper [5]. For FIR, we set its virtual cost to 0.01. Simulation results are summarized in Figs. 7–12, where the x-axis is the call-request number, or the number of call requests generated. Note that for each request number, 15 independent experiments (with different random seeds) are conducted. The results shown in the figures are the average values over 15 independent experiments.

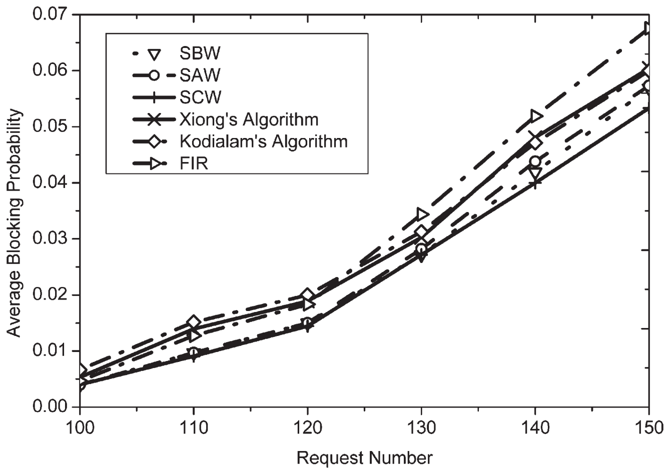

Fig. 7. Blocking probability versus request number (Topology I).

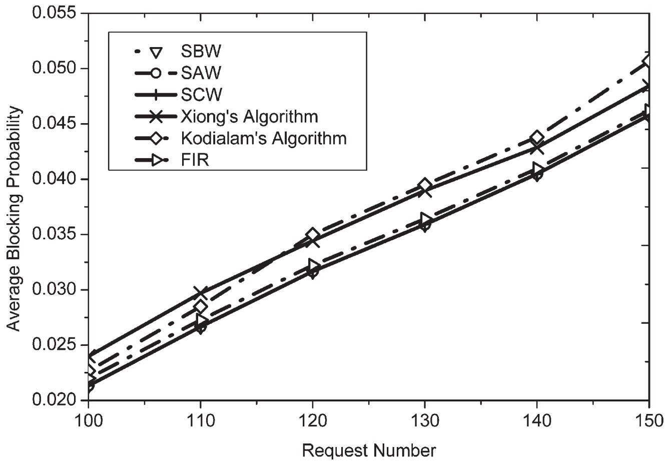

Fig. 8. Blocking probability versus request number (Topology II).

Figs. 7 and 8 are the graphs of blocking probability versus request number for the two topologies, respectively. From Fig. 7, we can see that the SCW algorithm gives the lowest call blocking probability. The SBW is a little better than SAW, but much better than Xiong's, Kodialam's, and FIR algorithms. In Fig. 8, the three two-step algorithms give comparable blocking performance, which is noticeably lower than that of Xiong's and Kodialam's algorithms, and marginally better than FIR.

[2]With this assumption, the impact of different call holding times on active and backup paths cannot be investigated. This is to the advantages of Kodialam's and Xiong's algorithms as they have ignored such impact.

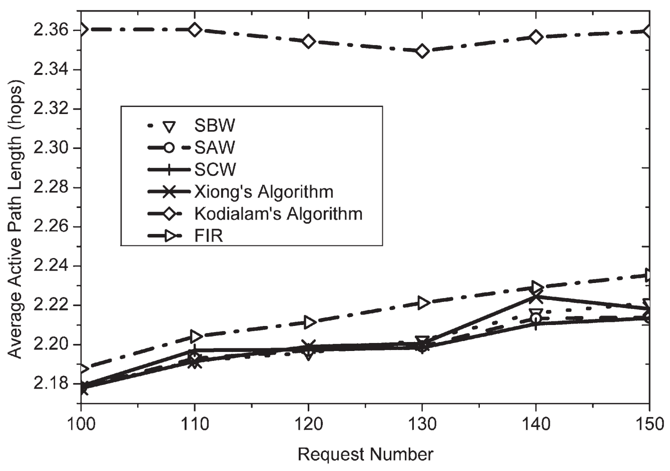

Fig. 9. Active-path length (hops) versus request number (Topology I).

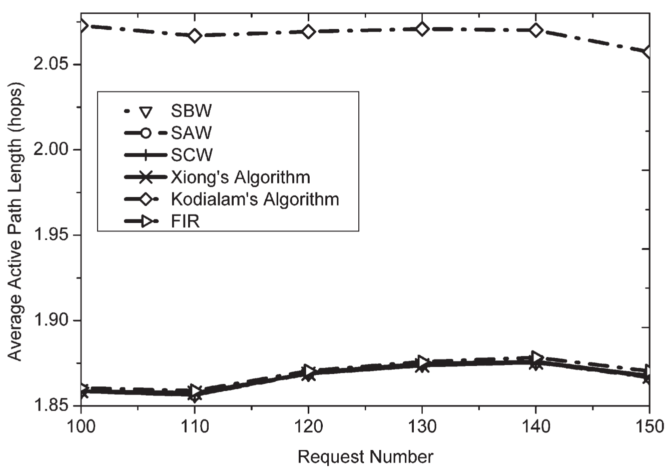

Fig. 10. Active-path length (hops) versus request number (Topology II).

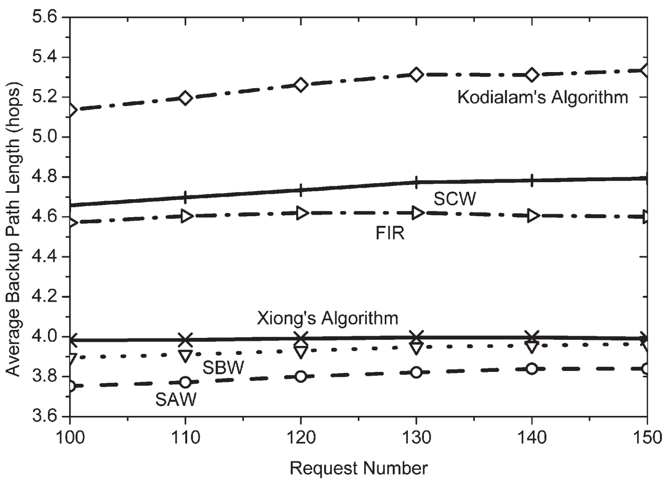

Fig. 11. Backup-path length (hops) versus request number (Topology I).

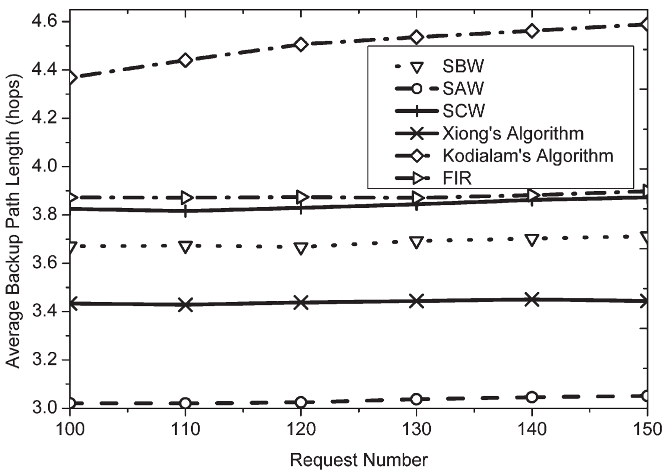

Fig. 12. Backup-path length (hops) versus request number (Topology II).

This effectively evidences that our two-step approach can indeed allow a network to admit more calls than the ILP approach and the simple shortest path pair method used by FIR. This is because our two-step approach can avoid using low capacity or near-saturated links, while the ILP approach or FIR only cares about the bandwidth cost.

Figs. 9 and 10 show the graphs of active-path length (in hops) versus request number. The active-path lengths of SBW, SAW, SCW, FIR, and Xiong's algorithms are almost the same, but remarkably better/shorter than Kodialam's algorithm. The poorer performance of Kodialam's algorithm is mainly due to its (unjustified) effort in minimizing the total unweighted bandwidth cost of active and backup paths. Xiong's algorithm does a better job because the weighting factor it adopted in the simulations ($\varepsilon = 0.1$) happens to be quite optimal. Actually, when $\varepsilon = 1$ (i.e., Kodialam's algorithm), the cost of an active path is generally much bigger than that of its backup path because of sharing. If $\varepsilon = 0.1$, the active path's cost is 10 times of its backup path, the selection of the backup path has a little chance of affecting or changing the selection of the active path. This effectively makes Xiong's algorithm (with $\varepsilon = 0.1$) function as a shortest path algorithm in determining active paths. Similarly, FIR also computes the shortest path for the active path. This explains why Xiong's algorithm (with $\varepsilon = 0.1$) and FIR can produce a rather short active-path length like our two-step approach.

From Figs. 11 and 12, which show the graphs of backup-path length (in hops) versus request number, the average backup-path length of SAW is remarkably shorter than Xiong's and Kodialam's algorithms. Although the lengths of SBW and SCW may be close to or even longer than Xiong's and FIR, they are still much shorter than Kodialam's algorithm. Note that the backup-path length of SAW can be adjusted by changing the value of parameter δ. When δ is small, the candidate set of backup paths gets larger and the backup-path length becomes shorter. For the SCW algorithm, the same procedure can be followed to adjust the value of the T threshold. This can also help SCW to reduce its backup-path length, if needed. We want to emphasize that it is extremely hard (if not impossible) to find a single routing algorithm that can achieve the lowest call blocking probability and the shortest active and backup paths at the same time. Reducing the length of the backup path may perhaps cause some performance degeneration on the other metrics. Our purpose of having three variants for backup-path routing is to provide a flexible and practical mechanism for network operators to choose what they prefer.

In simulating Kodialam's and Xiong's algorithms, a CPLEX 7.0 solver is used for finding the solutions of the corresponding ILPs. All of our simulation programs are run on a Sun Enterprise 6000 workstation. Considering the simulation of Topology II as an example, Kodialam's algorithm needs about 0.914 s to route a call, and Xiong's algorithm costs a little more than that, while our two-step series only take about 1.6 ms [the route computation time is collected from the central processing unit (CPU) execution time spent on routing algorithm subroutines]. As an on-demand routing algorithm, our two-step approach has a definite advantage over the ILP approach.

VI. Conclusion

In this paper, we have identified the problems with the popular integer linear programming (ILP) approach and full information restoration (FIR) approach of designing restorable dynamic quality of service (QoS) routing schemes. A novel alternative approach, called two-step restorable QoS routing, was then proposed based on two well-known QoS routing algorithms, widest shortest path (WSP) and shortest widest path (SWP). By properly exploiting their embedded features of minimizing resource consumption and load balancing, we can simultaneously maximize the bandwidth sharing among backup paths, and the network potential to admit future calls. Comparing with the best known existing algorithms, we showed that our two-step routing algorithms yield noticeably lower call blocking probability, shorter active-path length, and adjustable backup-path length. Besides, our two-step approach can find routes in a much shorter amount of time, which makes it more attractive for dynamic routing.

References

[1] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, RFC 3031, Jan. 2001.

[2] V. Sharma and F. Hellstrand, *Framework for Multi-Protocol Label Switching (MPLS)-Based Recovery*, RFC 3469, Feb. 2003.

[3] M. Kodialam and T. V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *Proc. IEEE Information Communications (INFOCOM)*, Anchorage, AK, 2001, vol. 1, pp. 376–385.

[4] M. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proc. IEEE Information Communications (INFOCOM)*, Tel Aviv, Israel, 2000, vol. 2, pp. 902–911.

[5] Y. Xiong, D. Xu, and C. Qiao, "Achieving fast and bandwidth-efficient shared-path protection," *J. Lightw. Technol.*, vol. 21, no. 2, pp. 365–371, Feb. 2003.

[6] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proc. IEEE Int. Conf. Network Protocols (ICNP)*, Atlanta, GA, Oct. 1997, pp. 191–202.

[7] M. Kodialam and T. V. Lakshman, "Restorable dynamic quality of service routing," *IEEE Commun. Mag.*, vol. 40, no. 6, pp. 72–81, Jun. 2002.

[8] Z. Wang and J. Crowcroft, "Quality of service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[9] G. Apostolopoulos *et al.*, *QoS Routing Mechanisms and OSPF Extensions*, RFC 2676, Aug. 1999.

[10] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 198–211, Feb. 2005.

[11] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, "Efficient distributed restoration path selection for shared mesh restoration," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 761–771, Oct. 2003.

[12] C. Qiao and D. Xu, "Distributed partial information management (DPIM) schemes for survivable networks—Part I," in *Proc. IEEE Information Communications (INFOCOM)*, New York, Jun. 2002, vol. 1, pp. 302–311.

Ji Li (S'04) received the B.Eng. degree in microelectronics and the M.Eng. degree in communication and information system from The University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2002, respectively. He is currently working toward the Ph.D. degree in computer networking at The University of Hong Kong, Hong Kong.

His research interests include network protection and restoration, and optical networks.

Kwan Lawrence Yeung (S'93–M'95–SM'99) received the B.Eng. and Ph.D. degrees in information engineering from The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, in 1992 and 1995, respectively.

He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, in July 2000, where he is currently an Associate Professor. His research interests include next-generation Internet, active queue management, packet switch/router design, all-optical networks, and wireless data networks.