A Two-Step Approach to Restorable Dynamic QoS Routing

Ji Li and Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong, PRC
{jili, kyeung}@eee.hku.hk

Abstract—**Aiming at minimizing the combined bandwidth cost of a pair of disjoint active and backup paths, a popular approach to designing *Restorable Dynamic QoS Routing* schemes is based on 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 a simple alternative approach, called *two-step* routing. In the first step, 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 and Composite-SWP. Combining both steps, three novel restorable routing algorithms, SBW, SAW and SCW, are obtained. Comparing with the existing best-known algorithms, we show that our two-step routing approach yields noticeably lower call blocking probability, shorter active path length, and adjustable 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 attractive for dynamic routing.**

I. INTRODUCTION

To deliver reliable service, networks such as Multi-Protocol Label Switching (MPLS) [1] require efficient recovery schemes to provide protection of the traffic carried on different data/active paths. The basic idea is that for each pair of communicating end-points, some *backup* paths are provisioned to protect the traffic carried on the *active* path. If the active path fails, transmission can be restored by rerouting the protected traffic to the backup paths. Under the assumption that there can be only one failure happens or exists at any given time, various recovery schemes [2-6] are designed to fully recover from such a single network fault. Those schemes differ from each other in some of the following ways, the speed of recovery, the amount of resources that must be pre-allocated (if any) to backup paths, the increased complexity of configuration and signaling, and the change in the length of data paths.

In this paper, we focus on the approach of end-to-end recovery. End-to-end recovery is also known as *path protection*. It is to use a single backup path to protect against any link or node fault on an active path. To achieve this, the backup path must be disjoint with the active path under its protection. We further assume that no prior knowledge about future call requests is available, and each new call request arrives with a pre-determined QoS/bandwidth requirement. When a new call arrives, an on-demand path computation, known as *restorable dynamic QoS routing* [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 request will be blocked.

In [7,10], several restorable dynamic QoS routing algorithms were proposed based on Integer Linear Programming (ILP) formulation. They aim at minimizing a combined bandwidth cost of a pair of disjoint active and backup paths. Since the resources of backup path can be shared, if more network state information is known, it is more favorable to efficient bandwidth sharing. Three network state information models were proposed in [7,10]: a) NS (No Sharing); b) SPI (Sharing with Partial Routing Information); and c) SCI (Sharing with Complete Routing Information). Without loss of generality, we focus on SCI model in this paper. In SCI model, the amount of bandwidth assigned to each individual active/backup path is assumed to be known at each new call arrival. For convenience, we call the SCI routing algorithm proposed in [7] as Kodialam's algorithm.

More recently, an enhanced (SCI) routing algorithm was proposed in [8], which is also based on ILP formulation. We call it Xiong's algorithm. It differs from Kodialam's algorithm in two major ways: 1) the objective function adopted is to minimize a *weighted* sum of bandwidth cost consumed by the pair of active and backup paths, in which the bandwidth cost of the backup path is weighted less; and 2) a non-zero backup bandwidth cost is introduced to a link which would otherwise have a zero cost in Kodialam's algorithm.

In this paper, we follow a simple two-step routing approach in designing restorable dynamic QoS routing. In the first step, 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 and Composite-SWP. Combining both steps, three novel restorable routing algorithms, SBW, SAW and SCW, are obtained. Comparing with both Kodialam's and Xiong's algorithms, we show that our two-step routing algorithms yield noticeably lower call blocking probability, shorter active path length, and adjustable backup path length (depending on the SWP variant adopted). Last but not the least, the running time of our two-step approach has a definite edge over the ILP approach, which makes it more attractive for dynamic routing.

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

II. Assumptions, Notations and Definitions

In this paper, we follow the common practice [7,8] of only requiring the backup path be *link disjoint* with the active path. This can be explained by the facts that co-located failover node devices can be easily installed for node protection. Assume each connection setup request arrives with a pre-determined bandwidth requirement w. For each link l, its physical bandwidth (BW) consists of three parts: ABW_l, BBW_l, and RBW_l. ABW_l is the total amount of reserved BW *dedicated* to all *A*ctive paths carried by link l. Such resources cannot be shared. BBW_l is the total BW occupied by all *B*ackup paths on link l. BBW_l can be shared by *some* backup paths, provided that their associated active paths are *disjoint*. Finally, the residual bandwidth RBW_l is the difference between the physical bandwidth of link l and the total consumed bandwidth ($\mathrm{ABW}_l + \mathrm{BBW}_l$).

We can see that for any future active path setup on link l, RBW_l is the only available BW can be used. If $\mathrm{RBW}_l \geq w$, the active path can be set up with a *link cost* equals to w. The total cost of setting up 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 active path a, the available bandwidth $\mathrm{RSW}_l(a)$ on link l consists of two components, residual bandwidth RBW_l, and the portion of BBW_l that can be shared to carry this backup path, denoted by $\gamma_l(a)$.

$$\mathrm{RSW}_l(a) = \gamma_l(a) + \mathrm{RBW}_l \qquad (1)$$

If $\mathrm{RSW}_l(a) \geq w$, the backup path can be set up. To encourage backup bandwidth resources sharing, the associated link cost is

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

Note that the link cost is infinite for those links traversed by active path a or with $\mathrm{RSW}_l(a) < w$. $\gamma_l(a)$ is always consumed with higher priority than RBW_l. In fact, RBW_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 BBW_l that is subject to share. Given two links m and l. Let A_m be the set of active paths carried on m. Let A_m^l be a subset of A_m that have their backup paths passing through link l. So from link l's point of view, active paths in A_m^l are not disjoint and thus they cannot share their reserved backup path resources on link l. Without loss of generality, let the total amount of backup bandwidth reserved for all active paths belong to A_m^l on link l be ξ_l^m. Then $\gamma_l(a)$ is given by

$$\gamma_l(a) = \mathrm{BBW}_l - \max_{m \in a} \xi_l^m . \qquad (3)$$

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, that are protected by the backup path on link l. Since $\max_{m \in a} \xi_l^m \leq \mathrm{BBW}_l$, $\gamma_l(a) \geq 0$.

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.

III. Two-Step Routing

We found that both Kodialam's [7] and Xiong's [8] algorithms can be improved if the following issues can be properly addressed.

- 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 to determine the optimal value for the weighting factor. In [8], the naïve trial-and-error approach is adopted.
- In both algorithms, 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.
- In Kodialam's algorithm, a backup path with zero cost will be chosen independent of its length. This results in long backup paths. In [8], it is shown to be a disadvantage in some cases. To address this problem, a non-zero cost is introduced to a link which would otherwise have a zero cost in Kodialam's algorithm. However, the way of determining the non-zero cost to be added is quite ad hoc.
- 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.
- 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.

In this paper, we propose to follow a two-step routing approach. In other words, 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. Since 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 widest-shortest path (WSP) algorithm is proposed for routing active paths in the first step, as detailed in Section III.A.

On the other hand, backup path is shared and will not be occupied unless there is a fault in the protected active paths, it is considered more important to facilitate load balancing rather than minimizing the reserved bandwidth. Shortest-widest path (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 *Basic-SWP* algorithm (Section III.B), *Approximate-SWP* algorithm (Section III.C), and *Composite-SWP* algorithm (Section III.D).

A. Widest-Shortest Path (WSP) Routing for Active Paths

With bandwidth saving as the primary objective and load balancing as the second, widest-shortest path (WSP) routing algorithm is chosen for active paths. If multiple shortest active paths between a source-destination pair are found, the one with the widest path bandwidth is chosen. Since RBW_l is the only available bandwidth for carrying active paths, the widest path is the path that has the largest bottleneck link residual bandwidth among all available paths from the source to the destination. In so doing, "shortest path" ensures that the resources consumed by the active path are minimized (first), and "widest shortest path" 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 RBW_l is less than w to produce an abridged topology. Then we apply the WSP algorithm to find the active path. WSP is implemented by modifying the Dijkstra's algorithm to find the shortest path with the widest bottleneck link RBW_l based on two metrics [9]: 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.

B. Basic Shortest-Widest Path (Basic-SWP) Routing for Backup Paths

For routing backup paths, we consider load balancing as the primary objective and minimizing backup path cost as the second. So shortest-widest path (SWP) routing [11] is chosen. With SWP, if there are multiple widest (backup) paths that are disjoint with the active path found using WSP in the first step, the (backup) 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 Eqns. (1) & (3). As a result, finding the set of "widest paths" first ensures load balancing, and finding the "shortest-widest path" next minimizes the reserved resources. We call this vanilla SWP adopted here as Basic-SWP.

Our implementation of the Basic-SWP follows that in [9]. First we prune the links for insufficient resources, i.e. links with $RSW_l(a) < w$. Then we apply Dijkstra's algorithm to find a widest path based on $RSW_l(a)$ defined in Eqn. (1). Let mRSW be the minimum of all $RSW_l(a)$ along the widest path found. We cut off the links with $RSW_l(a) <$ mRSW to generate another pruned topology. Finally we apply Dijikstra's algorithm again to find the shortest-hop distance path.

C. Approximate Shortest-Widest Path (Approximate-SWP) Routing for Backup Paths

A shortest-widest path may waste much bandwidth if it only considers links with available bandwidth higher than the required threshold mRSW. An example is shown in Fig. 1 for setting up a backup path with $w = 3$ between nodes 1 and 4. The number above each link is the total available bandwidth RSW_l 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 the Basic-SWP routing algorithm is used, mRSW 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 the Basic-SWP is *b*2, with a total path cost of 3 x 5 = 15. If we can lower the value of mRSW to 49, the same Basic-SWP will return the backup path *b*1, which has a much shorter length, 3 hops, and a much lower cost 9.

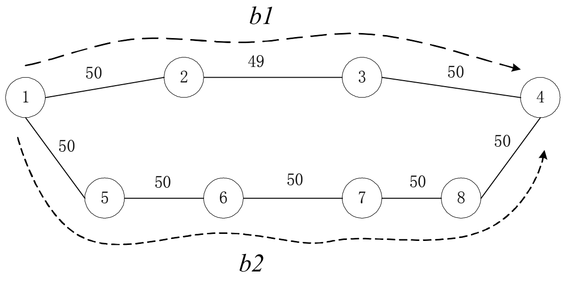

Figure 1. Approximate-SWP can yield a better performance. The number above each link is the total available bandwidth $RSW_l(a)$ of that link, not its link cost.

This implies that a slight relaxing on the "widest" path selection criterion (i.e. mRSW) can give a significant gain in backup path cost and length. Here we propose a variant of SWP, called *Approximate-SWP*, to capture this effect. In Approximate-SWP, a ratio δ ($0 \le \delta \le 1$) is defined to lower the mRSW threshold (by a factor of δ) to enlarge the set of candidate backup paths. When $\delta = 0$, Approximate-SWP degenerates into the simple shortest path algorithm. When $\delta = 1$, Approximate-SWP is the same as the Basic-SWP.

D. Composite Shortest-Widest Path (Composite-SWP) Routing for Backup Paths

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

Since the residual bandwidth component RBW_l can be used to admit both active and backup paths, it is more precious and should therefore consumed with lower priority than the other component $\gamma_l(a)$, 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, the "pruned network topology" is obtained by first removing links traversed by a or with $RSW_l(a) < w$, and then links whose $RSW_l(a)$ are less than mRSW (or δ*mRSW). 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[1] D_l, where

$$D_l = \frac{RBW_l}{1+\gamma_l(a)}. \qquad (4)$$

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

E. Three 2-Step Restorable Dynamic QoS Routing

Now we can combine WSP routing for active paths in the first step, with the three variants of the SWP routing for backup paths in the second step. This gives three two-step restorable dynamic QoS routing algorithms. We denote them by a three letter acronym following the convention of SxW, where the first "S" means the first step aims at finding a Shortest path, the last "W" means the second step tries to find a Widest path, and the middle "x" denotes the version of the WSP being used in the second step. So we have SBW, SAW and SCW, corresponding to the case that Basic-, Approximate-, or Composite-SWP is used in the second step.

IV. Simulation Results

The performance of the three two-step routing algorithms we proposed is compared with that of Kodialam's [7] and Xiong's [8] algorithms. 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.

Fig. 2 is the network topology we simulated, which is adopted from [7,10]. It consists of 15 nodes and 28 bi-directional links. There are two types of links: heavy links of 240 units of bandwidth in each direction, and light links of 60 units in each direction. Calls arrive one by one and call holding time is assumed long enough so we can consider accepted calls do not leave[2] [7,8,10]. The source and destination nodes of a call are picked up randomly. The guaranteed bandwidth of each call w is uniformly distributed between 3 and 8 units. For SAW algorithm, we set parameter δ of Approximate-SWP to 0.95. For Xiong's algorithm, we set its parameters $\varepsilon = 0.1$ and $\mu = 0.2$, as recommended in his paper. In Figs. 3-5, x-axis is the request number, or the number of call requests generated. For each request number, 15 independent experiments (with different random seeds) are conducted. The results shown in the figures are the average value over the 15 experiments.

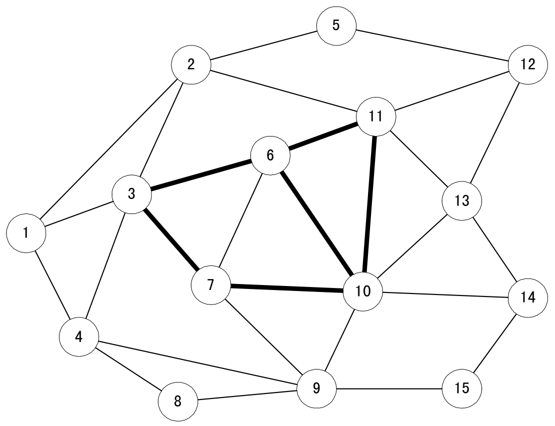

Figure 2. Simulation Topology

From Fig. 3 *Blocking Probability vs Request Number*, we can see that the SCW algorithm gives the lowest call blocking probability. SBW is slightly better than SAW, but much better than Xiong's or Kodialam's algorithm. This shows that our two-step approach can indeed allow a network to admit more calls than the ILP approach. For example, with 120 call requests, the call blocking probabilities for SCW, SBW, SAW, Xiong's and Kodialam's algorithms are 0.014, 0.015, 0.015, 0.019, 0.02 respectively. This is due to that ILP approach only considers the combined bandwidth cost, while two-step approach concerns load balancing as well as reducing bandwidth consumption.

In Fig. 4 *Active Path Length (in hops) vs Request Number*, the active path length of SBW, SAW, SCW and Xiong's are almost the same, but noticeable better/shorter than Kodialam's algorithm. The poorer performance of Kodialam's algorithm is mainly due to its 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 optimal. Since the bandwidth consumed by an active path is the product of its length and w, our two-step approach certainly works better in minimizing the resources dedicated to active paths than ILP approach.

From Fig. 5 *Backup Path Length (in hops) vs Request Number*, the average backup path lengths of SBW and SAW are remarkably shorter than Xiong's and Kodialam's algorithms. Although the length of SCW is longer than Xiong's, it is shorter than Kodialam's algorithm. Note that the backup path length of SBW 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 SCW algorithm, the same procedure can be followed to adjust the value of mRSW threshold. This can also help SCW to reduce its backup path length, if needed.

It should be emphasized that in simulating Kodialam and Xiong's algorithms, CPLEX 7.0 solver is used for finding the solutions of the corresponding ILPs. And all of our programs are run on Sun Enterprise 6000 workstation. For routing 150 requests, Kodialam's or Xiong's algorithm costs about 2 minutes to compute all the routes, while our two-step series only takes less than 0.3 seconds! (The time is measured on CPU

[1] It is possible to further fine-tune the distance expression in (4) for better performance.

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

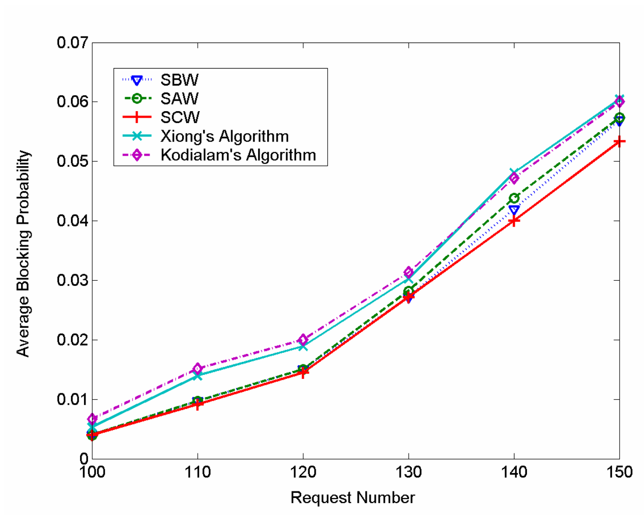

Figure 3. Blocking Probability vs Request Number

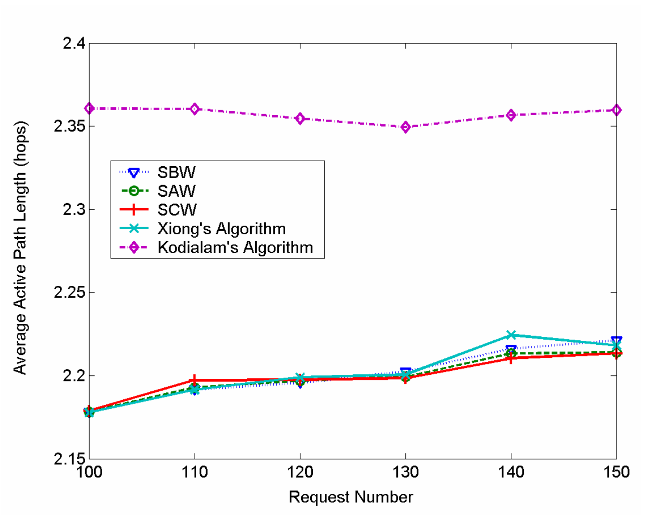

Figure 4. Active Path Length (hops) vs Request Number

time.) As an on-demand routing algorithm, our two-step approach has a definite edge over the ILP approach.

V. Conclusions

In this paper, we proposed a simple *two-step* routing approach for designing restorable dynamic QoS routing algorithms. It is 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 resources 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.

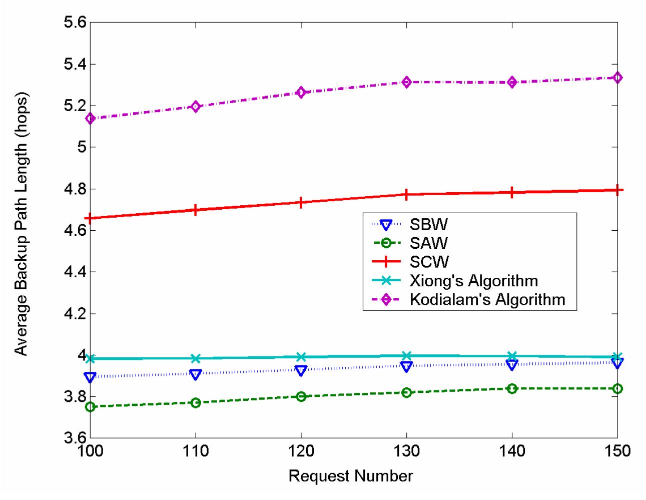

Figure 5. Backup Path Length (hops) vs Request Number

References

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

[2] C. Haung, V. Sharma, K. Owens and V. Makam, “Building Reliable MPLS Networks Using a Path Protection Mechanism,” IEEE Commun. Mag., pp. 156-162, Vol. 40, Issue 3, March 2002.

[3] V. Sharma and F. Hellstrand, “Framework for Multi-Protocol Label Switching (MPLS)-based Recovery” RFC 3469, Feb. 2003.

[4] A. Autenrieth and A. Kirstadter, “Engineering end-to-end IP resilience using resilience-differentiated QoS,” IEEE Communications Magazine, pp. 50-57, Volume: 40 Issue: 1, Jan. 2002.

[5] M. Kodialam and T.V. Lakshman, “Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information,” in Proc. IEEE INFOCOM 2001,Pages:376 - 385 vol.1

[6] Y. Bejerano, et al, "Algorithms for computing QoS paths with restoration," in Proc. IEEE INFOCOM 2003,Volume: 2, Pages:1435 - 1445

[7] M. Kodialam and T.V. Lakshman, “Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration,” in Proc. IEEE INFOCOM’00, 2000, pp.902–911.

[8] Y. Xiong, D. Xu and C. Qiao, “Achieving Fast and Bandwidth-Efficient Shared-Path Protection,” Journal of Lightwave Technology, VOL. 21, NO.2, Feb. 2003.

[9] Q. Ma and P. Steenkiste, “On Path Selection for Traffic with Bandwidth Guarantees,” Network Protocols, 1997. Proceedings., 1997 International Conference on , Oct. 1997

[10] M. Kodialam and T.V. Lakshman, “Restorable Dynamic Quality of Service Routing,” IEEE Communications Magazine, Volume 40, Jun 2002

[11] Z. Wang and J. Crowcroft, “Quality of Service Routing for Supporting Multimedia Application,” IEEE JSAC, 14(7):1288-1234, September 1996.