Improving Restoration Success in Mesh Optical Networks

Fang Yu[1], Rakesh Sinha[2], Dongmei Wang[3], Guangzhi Li[3], John Strand[2], Robert Doverspike[2], Charles Kalmanek[3], and Bruce Cortez[2]

[1]EECS Department, UC Berkeley, Berkeley, CA 94720

[2]AT&T, 200 Laurel Avenue South, Middletown, New Jersey 07748

[3]AT&T, 180 park avenue, Florham Park, NJ 07932

fyu@eecs.berkeley.edu, {sinha,mei,gli,rdd,crk,jls,}@research.att.com, bcortez@att.com

In shared mesh restoration, a distributed signaling protocol is used to reroute connections from failed service paths to restoration paths upon failure events. However, even when the network contains sufficient capacity, the restoration paths could be blocked for two different reasons: (1) with distributed restoration path selection schemes, multiple restoration paths may compete for the capacity of the same logical links, even when other logical links have sufficient capacity; (2) multiple restoration path set up attempts may compete for the same channels within the logical link (the "*glare* problem"), even when sufficient capacity is available within the logical link. This paper proposes a hybrid distributed/centralized restoration mechanism for restoration path selection and a channel selection scheme that eliminates almost all glares. As shown from simulation in a typical intercity backbone network, our proposed hybrid mechanism improves the first restoration attempt success ratio by 40% compared to a pure distributed restoration approach. In addition, the proposed restoration path selection algorithm saves up to 50% of restoration capacity compared with the disjoint shortest path algorithm and 15% compared with a previously published greedy algorithm.

1. Introduction

Many carriers are migrating away from SONET ring restoration for their core transport networks and replacing it with mesh restoration based on cross-connects (XC's). The specific architecture we will be considering uses XC's with STS-1 or STS-3 granularity electrical fabrics that are interconnected with OC-48 "links". These in turn are transported as wavelengths on fibers, each of which normally carries many wavelengths. There may be multiple links between two XC's; in this case the grouping of "similar" links (same XC end points, same fiber routing) is called a "logical link". The XC fabric allows the subdivision of each link into multiple TDM (STS-1) channels. If a connection requires multiple channels (e.g., an STS-12 if the XC's have STS-1 granularity), based on current technology, we assume that all must be provided in the same physical link but not necessarily in contiguous slots. Figure 1 illustrates a pair of XCs and a logical link containing two links connecting them. Each link contains a number of channels. In this architecture, fibers, wavelengths, and channels are pre-configured and static. Connections are routed over channels.

Both provisioning new connections and restoring existing connections after a failure require paths to be set up. Until recently this has been done using a **centralized** approach: A centralized server (e.g., a Network Management System -NMS) maintains a view of the entire network and is responsible for selecting paths and sending commands to the XCs to establish the connections. However, there is currently a trend towards distributed control (see, e.g., Sec. 5 in [19]), in which XC's implement a distributed control plane (e.g., GMPLS. In this approach, the source node of a connection selects a service path and a restoration path based on local information, and sets up the service path by sending signaling messages from the source node to other XCs along the path. The restoration path will be dynamically set up in case of a failure of the service path). In both the

centralized and **distributed** approaches, mesh restoration is typically provided via a restoration path that is fiber-disjoint from the service path [1].

The distributed approach typically distributes only routing information, while information about restoration path computations is normally not distributed, to keep signaling and state information simple [6]. As a result, each source node independently computes its restoration paths without knowledge of the restoration paths selected by other nodes. There are two pitfalls of this approach. If we were to just sum the capacities along all these independently computed restoration paths, the resulting capacity requirement would greatly exceed the optimal capacity requirement. On the other hand, if we do not adjust our capacity requirement to accommodate these restoration paths, multiple restoration paths can compete for the same bandwidth during restoration set up attempts, which will lead to blocking. When blocking occurs, the signaling setup request "cranks-back" to the source node [2] to try an alternative path, which increases the total restoration time. In this paper we propose and analyze a hybrid solution that utilizes a centralized restoration *path server* to optimize the restoration path selection, yet utilizes distributed control to compute service paths and set up service/restoration paths.

The centralized restoration path server could optimize restoration path selection. However the complexity of restoration requirements results in significant challenges to the designer of a path selection algorithm. For example, many carriers offer multiple levels of restoration priorities for services and install capacity in discrete units (such OC-48 to OC-192). Therefore, in this paper, we propose a simple, yet realistic path-selection heuristic that includes two levels of restoration priorities and minimizes the restoration capacity in the appropriate granularity. Previous published algorithms do not suit our objective since they commonly focus on minimizing the restoration capacity without considering restoration priorities and discrete bandwidth deployment [4,5,9,10].

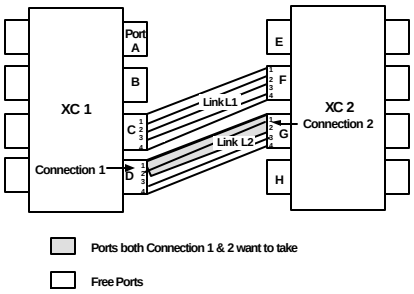

Figure 1: glare example

Even if we select the restoration paths judiciously, two connection set up attempts (traveling in opposite directions) may still compete for the same channels within a bi-directional logical link, even when the logical link contains sufficient capacity. Figure 1 shows an example, the first set-up for connection 1 reserves channel 1 and channel 2 of link L2 from XC1 to XC2. While this message is still in transit, another message traveling in the opposite direction for connection 2 reserves one or both of the same channels on the other endpoint of the link L2. Both messages, when they try to reserve the channels on their opposite endpoints, find that the channels have already been taken by another setup process and therefore fail. This lack of coordination is known as the **glare** problem. From our experience in the large network, up to 15% of the restoration attempts in the first try can fail due to glare. In this paper, we propose a new channel selection method to minimize glare while trying to minimize line-card bandwidth fragmentation so as to accommodate future large bandwidth requests.

This paper is organized as follows: Section 2 proposes a centralized/distributed restoration scheme and an efficient restoration path selection algorithm. Section 3 proposes a new channel selection scheme to reduce both glare and bandwidth fragmentation. We establish the benefits of our proposals through simulation experiments in a typical intercity backbone network in Section 4. Finally, conclusions are in Section 5.

2. Efficient Restoration Path Selection

As we mentioned before, in a distributed mesh optical network, it is possible that multiple restoration paths compete for limited channels in a logical link. In this section, we

propose a hybrid centralized/distributed approach to solve restoration blocking due to bad restoration path selection.

2. A Hybrid Restoration Approach

Our hybrid solution utilizes a centralized restoration *path server* or *path selection server* to optimize the restoration path selection, and a distributed signaling protocol to establish the paths. The objective is to achieve fast restoration upon network failure and more efficient use of network capacity. The connection provisioning procedure in this hybrid approach is as follows: when an ingress XC receives a connection request, it immediately selects and establishes the service path, then sends a restoration path request to the centralized path server along with the service path information. The path server returns an efficient restoration path based on the service priority, service path, and network status. If the path server is unable to select a restoration path due to insufficient capacity, it has the option to re-optimize the restoration paths of some or all previously established connections and then send new information to the source XCs. During a failure event, no communication exchange is required between XC and the path server and the restorations start simultaneously at different source XCs. This avoids the latency of centralized restoration.

To select the restoration paths, the centralized server should synchronize its network topology and state information such as link capacity and active service paths with the physical network. One solution is to integrate the restoration path selection server into a vendor-supplied EMS (Element Management System) that maintains this information. Then the distributed control plane is able to obtain a restoration path from the path server.

Since it might be computationally intensive to request an "optimized" restoration path from the centralized server when each connection is provisioned, an alternative to speed up the restoration path selection is to allow initial selection of a simple restoration path, such as disjoint shortest path, by the source XC. Then, in a regular cycle (e.g., daily), the centralized path server re-computes and downloads the restoration paths of some or all connections. Because there could be only few connections with sub-optimal paths between restoration path re-optimization events, such an approach might achieve virtually the same performance as optimal path selection, but without the real-time penalty of computing paths on a per-connection basis.

Such a hybrid restoration approach has several obvious advantages: First, the network achieves both the speed of distributed restoration and the use of optimized restoration paths. Second, this capability can be added to XC platforms currently using distributed restoration path selection with no or minimal change to existing signaling mechanisms. Third, the path server allows carriers to customize restoration to their own specifications, while avoiding modifications to distributed vendor XC solutions. Carriers can reflect restoration requirements that are not easily captured in vendor distributed path selection methods, such as a particular Shared Risk Group (SRG) topology [7] (e.g., shared fiber span or WDM structures), or service criteria (e.g., priority of connections).

2. B Restoration Path Selection Algorithm

Carriers may offer multiple levels of restoration priorities for different services. Suppose the network supports two types of connection services: Class 1 and Class 2. Class 1 services have more stringent restoration time requirements than Class 2 services. The inputs to the path selection algorithm include the network topology, service connections in STS-1 units and service type, as well as service paths for all connections. The output is an SRG-disjoint restoration path for each service connection and the corresponding network capacity-planning result with capacity purchased in units of OC-48.

In this section, we present a new heuristic (called *pushdown*) to achieve our objective. For performance comparison, we first discuss two commonly used algorithms for restoration path selection.

Shortest Restoration Path Selection

The shortest restoration path algorithm computes the shortest SRG-disjoint restoration path for a specified service path for each connection. If a SRG-disjoint restoration path is not available, it outputs a maximally SRG-disjoint shortest restoration path. This is achieved by temporarily assigning very large weights to logical links from the network topology that are part of the same SRG as logical links on service path. Then we run a shortest path algorithm, e.g., Dijkstra's algorithm, to compute a restoration path. The shortest path algorithm, by its very nature, will try to avoid links with large weights.

The capacity-planning algorithm considers one connection at a time and records the required bandwidth. For each link, an array $failneed_k[e]$ is maintained, giving the needed restoration capacity (in multiple of STS-1 units) on logical link k when SRG e fails. In order to provide 100% restoration for any single SRG failure, logical link k must support at least $M_k = \max_e failneed_k[e]$ STS-1s. If capacity is acquired in units of OC48, logical link k must be augmented with $\lceil M_k/48 \rceil$ OC48 units of additional capacity. For example, if M_k = 49 STS-1 units, then two OC-48s of capacity must be provisioned on logical link k. The shortest restoration path algorithm first computes the SRG disjoint restoration path for every connection. Then it updates the $failneed_k[e]$ of every logical link among the restoration path, then computes the maximum and outputs the $\lceil M_k/48 \rceil$ as the capacity-planning result. The running time for Dijkstra's algorithm using binary heaps is $O(E \log E)$, where E is the number of logical links. If we have C connections, the complexity of running Dijkstra's algorithm is $O(CE \log E)$ once for the service path and once for restoration path $\max_e failneed_k[e]$ takes an extra $O(R.E)$ steps where R is the number of SRGs. Thus the overall complexity is $O(RE + CE \log E)$

A Greedy Restoration Path Selection

Optimal SRG-disjoint restoration path selection is NP-hard so we need to resort to heuristics, such as the greedy online algorithm described in [6]. Unlike the shortest path algorithm, where each restoration path is computed solely based on its service path, this heuristic algorithm selects restoration paths to maximize the use of available (unused) bandwidth on existing links in the network. The intuition is that the capacity needed on any link is the maximum capacity needed in a single SRG failure. So if a SRG failures has already dictated high capacity requirement on a link, this extra capacity is treated as "free" in computing restoration paths for subsequent SRG failures. We consider one demand at a time, choosing the restoration path to minimize $\sum_k \lceil M_k/48 \rceil$ for each demand. It first sets the weight to infinity of each logical link that is part of the same SRG as the service path for the demand. Then, it sets the weights of each remaining logical link with sufficient unused bandwidth to accommodate this logical link to a low value, and sets the weights of logical links with insufficient unused bandwidth to a high value. (A logical link may have unused bandwidth because of the rounding up of the $\lceil M_k/48 \rceil$ or restoration capacity for other restoration paths.) After setting all the weights, it runs Dijkstra's algorithm to select a restoration path using minimal additional OC-48's needed. Similar to the shortest path algorithm, after updating the $failneed_k[e]$ of every logical link among the all the restoration paths, it computes the maximum and outputs the $\lceil M_k/48 \rceil$ as the capacity-planning result.

For each connection, we modify the *failneed* array and compute a Dijkstra's algorithm. The worst case complexity of modifying the *failneed* array is equal to its size, $O(RE)$. Thus the overall complexity is $O(CRE + CE \log E)$ overall.

Pushdown Restoration Path Selection

We propose the *pushdown* path selection algorithm that considers two classes of connections, where class 1 services have the more stringent restoration time requirement. Since the time to establish a restoration path depends on the number of XCs on the path, the pushdown algorithm selects the shortest SRG-disjoint path as the restoration paths for

Class 1 connections. For Class 2 connections, the pushdown algorithm selects an SRG-disjoint path, while attempting to minimize the total number of OC-48s.

The pushdown algorithm operates in two phases. The first phase is based on the heuristic algorithm: we consider one service connection at a time and apply a greedy algorithm same as the algorithms discussed above. In the second phase, we try to do a global optimization. In order to provide 100% restoration, if M_k = 49 STS-1 units, then two OC-48s of capacity must be provisioned on logical link k. Recall that M_k is the maximum requirement among all SRG failures. It is possible that only one SRG failure requires two OC-48s and all the remaining SRG failures require only one OC-48 of capacity. Thus, we try to select alternative restoration paths for some of connections with a goal of bringing M_k down to one OC-48. We achieve this by first reducing one OC-48 on logical link k and fixing the required OC-48s on all other logical links. As a result, a few connections, which required two OC-48s on logical link k, may no longer have a restoration path available. Then we try to select alternative restoration paths for these connections without increasing the required capacity on any other logical links. If we succeed, we push down one OC-48. If we fail, we restore the original restoration paths and repeat this process with the next logical link. We iterate until no OC-48 can be pushed down.

As analyzed in the case of greedy restoration path, the complexity of the first phase is $O(CRE + CElogE)$. The complexity of the second phase is proportional to the number of "pushed-down" links, and each pushdown requires finding alternate paths for a few connections. The total complexity of the second phase is $O(P\ ElogE)$ where P is the product of (number of pushed-down links) and (average number of path computations per pushed-down link). In a typical experiment, P is of the same order as C, the number of connections. So the second phase at most doubles the running time. In our experiments, this heuristic ran for a couple of minutes on a large Intercity backbone network. The pushdown algorithm may not generate the optimal routes for all connections. However, since it is a simple heuristic, restoration paths can be computed very quickly. Thus, it is well suited for the centralized path server to periodically re-compute the restoration paths and load them to XCs.

3. Efficient Link/Channel Selection

We have used the term "path" to refer to the selection of a sequence of logical links, but not the selection of the specific link and channels to use within each logical link, which is most efficiently done by the XC's when they receive a path setup signalling message. Due to the limitation of current technology, we assume all the channels for a connection must be provided in the same physical link but not necessarily in contiguous slots. This problem is similar in spirit to the one-dimensional bin-packing problem [14,15,16,17], and routing sublamda connection requests at the logical topology in an IP-over-optical network [19,20,21,22], however it is more difficult because it involves simultaneous channel selection from both ends of the link. To avoid glare, we need to arbitrate the process so that different setup attempts do not compete for same channel. Let's first describe two well-known schemes for link selection: *Best-fit* and *Hi-Lo*.

Best-fit link selection is an algorithm to minimize the bandwidth fragmentation. Suppose there are K links between two XCs. Each link i has available bandwidth $aval[i]$. When a connection request arrives of size b bandwidth units, select the lowest indexed link, j, with $aval[j] \geq b$. In this manner, the large contiguous channels on other links are intact and thereby reduce fragmentation. However, when there are connection requests from both end of the link simultaneously, choosing the minimal available link would greatly enhance the chances of both ends selecting the same link and causing a glare.

Hi-Lo link selection aims to minimize glare. The basic idea of Hi-Lo is: between any pair of XCs connected by one or more links, pre-select one of them as "Hi XC" and the other as "Lo XC". We also predetermine an ordering of the links. The Hi XC selects the

highest indexed link, j, with $aval[j] \geq b$. The Lo XC selects the lowest indexed link, k, with $aval[k] \geq b$. Glare only happens if two connections are using up most of the channels in the all the links. Note that the Hi-Lo XC selection involved in this scheme is different from the master-slave scheme described before: It does not require any signaling extensions and both XCs can make channel selection decisions at the same time. The selection of which XC should be Hi can be simply based on the higher XC id.

For initial provisioning, because of the longer time scale, it is not usually subject to contention among multiple connections, and therefore, glare is not an issue. Then the main consideration is to keep the fragmentation rate low in order to accommodate future requests of large sizes. Thus we recommend using a "Best-fit" link selection algorithm.

During restoration, however, we can't use best-fit scheme because glare is a serious problem. A node or fiber span failure may cause the reroute of a large number of connections in a small time frame, e.g., under 100ms, thereby creating a lot of contention. In addition, restoration has a stringent time scale, so we can't rely on crank-back mechanism to try multiple attempts.

One way to avoid glare is to use a master-slave relationship: for each pair of neighboring XC interfaces, select one XC interface as "master" and the other as "slave". The master XC interface is responsible for selecting channels for connections traveling in either direction. If the connection is traveling from master towards the slave node, the master node can select links and channels and convey this to the slave node. For connections traveling from slave to the master node, an extra message conveying the request from slave to master and another message conveying the link/channel selection from master to slave are needed. Because of this additional signaling overhead, many equipment providers do not support this.

Before we introduce our new proposal in detail, let's look at an example of how the free capacity of different links between a pair of XCs might evolve during the provisioning phase. For simplicity we assume that all links have the same initial available capacity (In fact our scheme can be generalized as long as we know the initial distribution of the available capacities.) For the first provisioning request, all links provide identical packing so it picks the first link. Now the first link has the smallest available capacity. By the property of Best-fit algorithms, the next several connections will all go to the first link. When the available capacity of the first link can't support a new request, this provisioning request needs to go to the second link. The next several connections will be distributed among the first two links and so on. At the end of all connection provisioning, the available capacity of links will be roughly an increasing function of link index. It is highly likely that the highest indexed links have no connections on them.

For connection restoration, if we run Hi-lo scheme at this stage to avoid glare, we will fragment those high indexed links. So, here we propose an ***Interleave scheme*** to accommodate all connections in the first few links while still avoiding contention. The intuition behind the Interleave scheme is to run a mixed best-fit and Hi-Lo methods on a modified ordering of links, where both nodes are likely to select among the first few links. Thus we order the links as 1,3, 5, …, M, N,…,6,4,2, where $M = 2*\lceil K/2 \rceil - 1$ and $N = 2*\lfloor K/2 \rfloor$. The Interleave scheme pre-selects one XC as "Even" and the other as "Odd". The "Odd-XC" examines the links {1,3,5,..,M} and selects the link with the smallest available capacity that can still accommodate this request. If it cannot find a link, it examines the links in the order of {N ,...6,4,2} and selects the *first* link that can accommodate the request. The "Even-XC" examines the links {2,4,6, …, N } and selects the link with the smallest available capacity that can still accommodate this request. If we can not find a link, it examines the links in the order of {M ,...,5,3,1} and selects the *first* link that can accommodate the request. As a result, the two XCs consider links in opposite order while making a conscious effort to avoid high-indexed links.

The interleave scheme roughly simulates two Best-fit selections among the low-indexed links. The advantages of the Interleave over Hi-Lo lies in the assumption that the Best-fit link selection scheme has been used during the service provisioning phase, which leaves available bandwidths in a certain order. This observation can be generalized. As long as we know the link selection mechanism used during the service provisioning phase and the expected distribution of available bandwidths, we can tailor the link selection mechanism during restoration to reduce fragmentation in addition to reducing glare. This scheme does not require any signaling extensions. However, as shown in the simulation section, it can eliminate glare in most cases.

4. Performance Evaluation

To evaluate the performance of proposed schemes, we implemented a GMPLS-like protocol simulator and compared the restoration time and restoration success rate with other publicly available schemes in a typical intercity backbone network and demand forecast. This network, with 95 XCs and 164 logical-links, and the demand forecast have been used in other studies [8,12]. In our simulation, a logical link consists of multiple OC-48 links. We assume requests for bandwidth in units of STS-1, STS-3, STS-12, or STS-48. The XCs are assumed to have STS-1 granularity. The demand set is randomly generated among source-destination pairs with different bandwidth units. For failure scenarios, we considered the Shared Risk Groups (SRG [7]) and identified 160 SRGs. We simulated all the failure cases and analyzed the restoration process under each failure scenario. All the simulation results reported here are the averages of all SRG failures.

4.A. Efficiency of Pushdown Algorithm

First, we compare the pushdown algorithm with disjoint shortest restoration paths and the greedy algorithm described in section 2. For every demand set, we calculate the total number of OC-48s required for each restoration scheme, then compute the restoration overbuild, which is the ratio of total number of OC-48s for restoration to total number of OC-48s for service provisioning. The restoration overbuild considers only single SRG failure. Figure 2 shows the restoration overbuilds for different demand sets. The top curve is the overbuild requirement for the SRG-disjoint shortest restoration path. The middle curve is the overbuild requirement using the greedy algorithm provided in paper [6]. As shown, the pushdown algorithm reduces the restoration overbuild up to 50% compared with the shortest path algorithm, and 15% compared with the greedy algorithm.

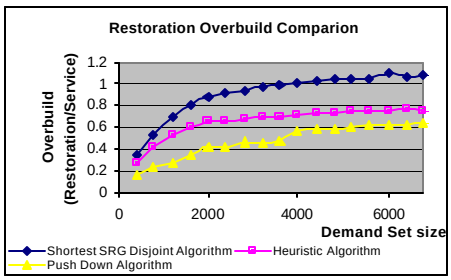

Figure 2: restoration overbuild

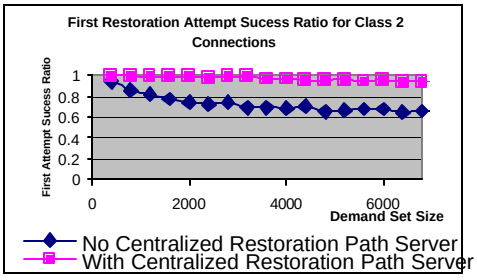

Figure 3: first attempt success ratio for class 2

4.B Hybrid Restoration Path Setup Algorithm

To evaluate restoration process efficiency, we compared our proposed hybrid approach with a fully distributed approach (using shortest SRG disjoint restoration paths) with the same network and demand matrix models. We used the pushdown algorithm to plan the capacity and evaluated the restoration success ratio by simulation of the distributed restoration process with interleaved link selection scheme. Both approaches restore nearly 100% of Class 1 connections in the first attempt since class2 connections are delayed so there is enough bandwidth for Class 1 connection. Figure 3 shows the results

for Class 2 connections. Our approach restores nearly 100% of Class 2 connections in the first attempt. In contrast, the distributed approach can only restore 60% of Class 2 connections at first attempt under high demand load, and the blocking is mainly due to multiple restoration paths competing insufficient logical link capacity although there are sufficient capacities in other logical links. This shows that restoration path selection is an important factor to improve restoration success.

4.C Link Selection Algorithms

To evaluate the impact of link selection algorithms, we ran two types of simulations on performances of various link selection algorithms: on a single logical link and on an intercity backbone network. We use three figures of merits to study the efficiency of the link selection algorithms: *bandwidth of rejected connections due to fragmentation*, *bandwidth of rejected connections due to glare*, and the *total failed bandwidth*. Note that the first two are mutually exclusive. When a connection gets rejected at the ingress interface of a XC, there are two possibilities. If the cumulative available bandwidth of the logical link cannot accommodate this connection, this is a result of bad capacity planning or poor routing decisions. This situation never arises in our simulation because we use the hybrid approach for capacity planning and restoration path selection. The other possibility is that the cumulative bandwidth is enough to accommodate this connection but is fragmented across multiple links. In this case, we mark this rejection as resulting from "bandwidth fragmentation." If the connection gets accepted at the ingress interface but then gets rejected at the other endpoint of this logical link, we mark this rejection as resulting from "glare". To evaluate the overall efficiency of an algorithm, we add the above two cases to get the *total failed bandwidth*.

4.C.1 Simulation on Single Logical Link

Each logical link may consist of multiple OC-48 links. During the simulation, we keep generating the requests until the total bandwidth of the requests exceeds the total logical link capacity and we discard the last request. Among these requests, we select the first 80% as service path setup requests and set them up according to best-fit link selection algorithm. The remaining 20% are taken as restoration requests. We generate less restoration request than service request because typical failure in the network won't fail all service connections. There are other three parameters that may affect the performance: *communication delay*, *processing delay*, and *average arrival interval*. Communication delay denotes the time stated as the time from when a message is put on the output queue at one XC until it is handed to the processing software at the receiving XC. Processing delay is the time used to process each request. Average arrival interval records the time difference between two adjacent restoration requests. In the following simulation, we test the performance of three link selection algorithms under different scenarios. For each scenario, we report the average of 1000 independent runs.

Different Distribution of Requests:

In this scenario, we generate request with different distribution of requests. The parameter values we used are: communication delay = 3 *ms*, average arrival interval= 0, processing time = 1 *ms*, logical link size =10 OC-48s. Tables 1,2,3 represent the bandwidth of failed connections due to glare, fragmentation, and both respectively. In all three tables, the first column shows the request distribution pattern. For example, 1:1:1:1:1 means all five bandwidth requests have the same probability, and 3:1:1:1:1 means requests with STS-1 bandwidth are 3 times more than STS-3 requests. From Table 1 it is not surprising that the Best-fit algorithm has the largest number of glares as explained in section 3.B. Both Hi-lo and Interleave schemes have low glare rates, because connection from two sides will not select the same link unless necessary to accommodate both requests. Interleave results in less failure caused by glare than Hi-lo. This is because the interleave scheme creates less fragmentation, so the chances that only one link can accommodate the request are decreased. During the simulation, we generated just enough

requests to fill the capacity of all links. If there is no fragmentation, all the connection should be able to find a link that has enough remaining bandwidth for it. Table 2 records the total bandwidth of failed connection due to fragmentation. In our simulations, Best-fit does not reject any connections due to fragmentation, while both interleave and Hi-lo cause fragmentations. When there are higher portion of large requests (last two rows), Interleave performs much better than Hi-Lo. Table 3 presents total failed bandwidth either due to glare or fragmentation. It is clear that Best-fit scheme performs the worst while Interleave scheme performs the best in all distributions.

	Table 1		
STS-1: STS-3: STS-12: STS-24: STS-48	Best-fit	Hi-Lo	Interleave
1:1:1:1:1	31.26	16.61	14.12
3:1:1:1:1	30.14	13.23	11.42
1:3:1:1:1	31.03	14.04	11.57
1:1:3:1:1	35.97	12.24	11.36
1:1:1:3:1	35.46	13.33	10.53
1:1:1:1:3	28.57	19.29	14.21

Table 2		
Best-fit	Hi-Lo	Interleave
0	1.30	0.80
0	3.96	3.54
0	2.82	2.78
0	0.6	0.58
0	0.34	0.19
0	2.35	1.34

Table 3		
Best-fit	Hi-Lo	Interleave
31.26	17.91	14.93
30.18	17.19	14.96
31.03	16.86	14.35
35.97	12.84	11.94
35.46	13.66	10.72
28.57	21.64	15.56

Table 1, 2, 3. Bandwidth (units of STS-1) of failed connections due to glare (Table 1), fragmentation (Table 2) and both (Table 3).

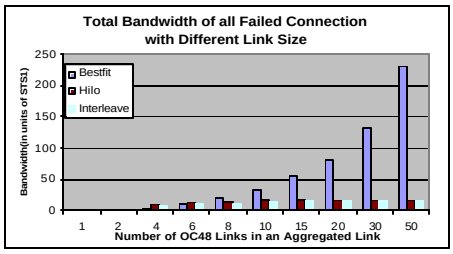

Figure 4: total bandwidth of failed requests

Different Size of Logical link:

In this simulation, we vary the number of OC-48 links in a logical link from 1 to 50. Similar to previous simulation, we set the communication delay to 3 *ms*, average arrival interval to 0 *ms*, and processing time to 1 *ms*. The distribution of requests in all categories is 1:1:1:1:1. Figure 4 shows the total bandwidth of failed requests. Clearly, the Interleave and Hi-lo schemes greatly outperform the widely used Best-fit algorithm. In all the cases, interleave behaves no worse than Hi-Lo. When a logical link has medium size (8-15), a typical size in today's inter-city transport networks, the Interleave scheme reduces an average of 14% of failed bandwidth compared to Hi-lo scheme.

Different Arrival Interval and Communication Delay

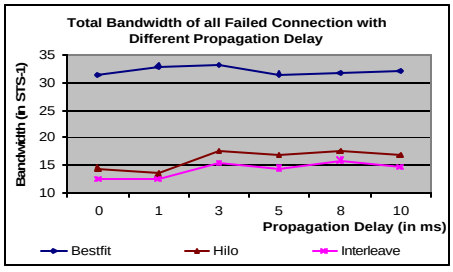

Figure 5: Impact of average arrival rate

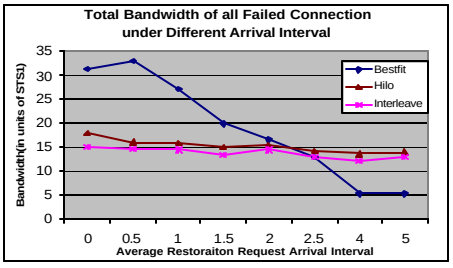

Figure 6: Impact of communication delay

In previous simulations, we vary the average arrival interval from 0 to 5 *ms* and the other parameters are set as the same as previous simulation. The logical link is set to be 10 OC48s. The total bandwidths of failed connections are showed in Figure 5. Best-fit performs poorly when the average arrival interval is small because of glare. However,

when the average arrival intervals increase, Best-fit out-performs interleave because in this case, the chances of two connections coming to different end of the link simultaneously becomes much lower. Then the chance of glare is dramatically decreased. This is similar to the connection provisioning where Best-fit is the best choice. Overall, the performance of Interleave is stable under all arrival intervals. Figure 6 illustrates the impact of communication delay with average arrival interval of 0 and l other parameters the same. Communication delay doesn't have too much impact on bandwidth of failed connections. Interleave scheme stays the best for all values of communication delay.

4.C.2 Simulation on an Intercity Network

	Fragmentation	Glare	Total
Best-fit	7408	31632	39040
Hi-Lo	17820	2170	19990
Interleave	17534	2244	19778

Table 4 – Bandwidth (units of STS-1) of all failed connections

Table 4 shows simulation results for the three link selection schemes in an intercity backbone network combined with hybrid restoration scheme and pushdown restoration path selection. It is clear that the Interleave link selection can greatly reduce glare compared to Best-fit. It does create more fragmentation, but it can still reduce 50% overall bandwidth of failed connection compared to Best-fit algorithm. Interleave algorithm performs better than Hi-lo, but the improvement is not dramatic. This is because in the current network, small connections such as STS-1 and STS-3 dominate. Thus, fragmentation will not be a serious issue. Moreover, the average number of links in a logical link is small, which does not allow us to differentiate between different link-selection algorithms. The advantage of Interleave grows with larger networks and greater proportion of high rate connections (as shown in the Table 2).

5. Conclusion

In this paper, we studied restoration path-setup blocking in a distributed restoration process. Specifically, we studied contention due to uncoordinated restoration path computation and glare due to uncoordinated link selection even if there is enough capacity. We proposed two schemes for reducing blocking so as to improve the restoration success rate. The first one is a hybrid distributed/centralized approach that combines the merits of centralized and distributed solutions. It avoids the scalability issues of centralized solutions by using a distributed control plane for service path computation and service/restoration path provisioning. The hybrid approach improves the first restoration attempt success rate by 40% compared with pure distributed approach. We also presented a restoration path computation algorithm. Simulation results showed that our algorithm saved up to 50% of restoration capacity compared with the shortest path algorithm and 15% compared with a previously published greedy algorithm. The second one is a link selection algorithm to reduce glare. This scheme only requires a minor change to most XCs currently using maximal packing algorithm, however, it can eliminate glare in most cases.

Reference

1. G. R. Ash. "Dynamic Routing in Telecommunications Networks," *McGraw-Hill*, (1997).
2. G. Li,C. Kalmanek, and R. Doverspike, "Fiber Span Failure Protection in Mesh Optical Networks," *Optical Networks Magazine,(2002)*.
3. E. Modiano, "WDM-Based Packet Networks," *IEEE Comms Magazine*, ,March 1999, pp. 130-135.
4. D. Stamatelakis and W. D. Grover, "IP Layer Restoration and Network Planning Based on Virtual Protection Cycle", *IEEE JSAC Special Issue on Protocols and Architectures for Next Generation Optical WDM Networks*, vol.18, no.10, (Oct. 2000).

5. M. Kodialam and T. V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information", *IEEE INFOCOM*, (Apr. 2001).
6. G. Li, D. Wang, C. Kalmanek, and R. Doverspike., "Efficient distributed path selection for shared restoration connections," in *Proceedings of IEEE INFOCOM*, (June 2002).
7. J. L. Strand, J.; A. L. Chiu, , R. Tkach, . "Issues For Routing In The Optical Layer", in *Proceedings of IEEE Communications Magazine*, 2/2001, vol. 39, no. 2, pp. 81 –87.
8. F. Yu, R. Sinha, R.D. Doverspike, B. Cortez, J. Strand, "Link Selection Schemes for Avoiding Channel Contention", in *Proceedings of Fourth International Workshop on the Design of Reliable Communication Networks*, (October 2003).
9. M. Kodialam and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration", in *Proceedings of IEEE INFOCOM*, March 2000.
10. R. Doverspike, *et al*, "Fast restoration in a mesh network of optical XCs", in *Proceedings of Optical Fiber Communication conference,(19*99).
11. E. Bouillet, *et al*., "Enhanced Algorithm Cost Model to Control Tradeoff in Provisioning Shared Mesh Restored Lightpaths", in *Proceedings of Optical Fiber Communication conference (*2002).
12. F. Yu, D. Wang, R.Sinha, G. Li, B. Doverspike, and C. Kalmanek, "Hybrid Centralized/Distributed Approach to Optical Network Restoration," in *Proceedings of Optical Fiber Communication conference*, (2003).
13. G. Li, J. Yates, R. Doverspike and D. Wang, "Experiments in Fast Restoration using GMPLS in Optical / Electronic Mesh Networks" *Post-deadline,* in *Proceedings of Optical Fiber Communication conference* , (2001).
14. J. Csirik1 and D. S. Johnson, "Bounded Space On-Line Bin Packing: Best is Better than First", *ALGORITHMICA* 31:2, (2001), pp 115-138,.
15. E.G. Coffman, Jr, *et al*, "Approximation Algorithms for Bin-Packing -- An Updated Survey", *In Algorithm Design for Computer System Design*, (1984).
16. D. S. Johnson, "Fast Algorithms for Bin Packing", *Journal of Computer and System Sciences* 8, (1974), pp 272-314.
17. C. Kenyon, "Best-fit bin-packing with random order", *In the proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, (1996).
18. J. L. Strand, "Optical Network Architecture Evolution", chapter in I. Kaminow and T. Li (eds.), *Optical Fiber Telecommunications IV*, Academic Press, (2002).
19. K. Zhu, B. Mukherjee, "Traffic Grooming in an Optical WDM Mesh Network", *in the proceedings of* IEEE ICC 01, (June 2001).
20. C. Assi, Y. Ye, A. Shami and M. A. Ali, "Integrated Routing Algorithms for Provisioning Sub-Wavelength Connections in IP-Over-WDM Networks," *Journal of Photonics Networks*, (June 2002).
21. Y. Ye, C. Assi, S. Dixit, and M. A. Ali, "Simple Dynamic Integrated Provisioning/Protection Scheme in IP over WDM Networks," *IEEE Communication Magazine*, Nov. 2001, Vol. 39 No. 11, PP. 174-182.
22. C. Assi, Y. Ye, A. Shami, S. Dixit, and M. A. Ali, "On the Merit of IP/MPLS Protection/Restoration in IP over WDM networks," *in the proceedings of* IEEE GLOBECOM 01, San Antonio, Texas.