Survivable Virtual Concatenation for Data Over SONET/SDH in Optical Transport Networks

Canhui (Sam) Ou, *Student Member, IEEE*, Laxman H. Sahasrabuddhe, Keyao Zhu, Charles U. Martel, and Biswanath Mukherjee, *Member, IEEE*

***Abstract*—Next-generation SONET/SDH technologies—namely, generic framing procedure, virtual concatenation, and link-capacity-adjustment scheme—enable network operators to provide integrated data and voice services over their legacy SONET/SDH infrastructure to generate new revenue. An important open research problem on data over SONET/SDH (DoS) is survivability: SONET automatic protection switching is too resource inefficient for data services, and the protection mechanisms of data networks are too slow for mission-critical applications.**

We propose two approaches for provisioning survivable DoS connections. Our approaches exploit the tradeoff between resource overbuild and fault-recovery time while utilizing the inverse-multiplexing capability of virtual concatenation to increase backup sharing. Our results show that one approach achieves low resource overbuild and much faster fault recovery than that of data networks, and the other approach achieves fast fault recovery comparable to SONET 50-ms protection (for typical U.S. backbone networks) while still achieving modest backup sharing. We further investigate the tradeoff between network blocking performance and network control and management complexity resulting from the number of paths M a connection can be inversely multiplexed onto: larger M leads to more freedom in routing and better network performance but increases network control and management complexity. Our results indicate that the network blocking performance for small values of M (e.g., $M = 2$ for some representative backbone network topologies) is almost as good as the case in which M is infinity.

***Index Terms*—Data over SONET, next-generation SONET/SDH, survivability, virtual concatenation, WDM.**

I. Introduction

A. Next-Generation SONET/SDH Technologies

SONET/SDH has historically been the dominant transport infrastructure optimized for reliable delivery of voice and private-line services in metro and backbone networks. It likely will remain in the foreseeable future as the dominant framing layer for supporting integrated data and voice services over optical transport networks to leverage the existing SONET/SDH telecom infrastructure because of the emerging data-over-SONET/SDH (DoS) technologies, namely, generic framing procedure [17], virtual concatenation [1], [4], and link-capacity-adjustment scheme [3].

Generic framing procedure (GFP) is a traffic-adaptation protocol which maps either a physical-layer signal or a logical-link-layer signal to an octet-synchronous signal such as a SONET/SDH channel. While data traffic can be first mapped into SONET/SDH signals via GFP and then transported over a SONET/SDH network, such an approach can lead to significant bandwidth inefficiency as traditional SONET/SDH signal hierarchy provides a rigid, tiered bandwidth allocation, e.g., STS-1, STS-3c, STS-12c, STS-48c, and STS-192c. For example, a STS-48c SONET container (of approximate capacity 2.5 Gb/s) is needed to carry a Gigabit data connection, such as a Gigabit Ethernet line, in such a network. This results in about 60% bandwidth wastage. Clearly, flexible bandwidth-allocation mechanisms are needed.

Virtual concatenation (VC) is one such mechanism. VC is an inverse-multiplexing technique which groups an arbitrary number of SONET/SDH containers, which may not be contiguous, to create a bigger container. A network operator can combine any number of either low-order containers (VT1.5s/VT2s in SONET) or high-order containers (STS-1s/STS-3cs in SONET), depending on the switching granularity, to create one VC group (VCG). With VC, it is now possible to provide fine granular bandwidth, e.g., multiples of VT1.5 (payload capacity about 1.536 Mb/s) for low-order VC and multiples of STS-1 (payload capacity about 48.384 Mb/s) for high-order VC. For example, 21 virtually concatenated STS-1s, denoted by STS-1-21v, can be used to provision a Gigabit data connection. Compared to the STS-48c bandwidth needed to carry this connection without VC, only STS-21 bandwidth is needed with VC, resulting in a huge capacity savings (of approximately 60%).

A connection carried by a VCG, referred to as a VC connection, can be inversely multiplexed onto multiple paths at the source node and merged at the destination node. Each path of a VCG, referred to as a VCG member, is routed across the network independently. As a result, VC works across traditional SONET/SDH networks because only the source and destination nodes are aware of the VCG. Since a connection can be inversely multiplexed onto multiple paths, network load can be distributed more evenly and network performance (or information-carrying capacity) can be improved. One

Manuscript received January 31, 2004; revised February 1, 2005. This work was supported by the National Science Foundation under NSF Grants ANI-98-05285 and CCR-85961. A summarized version of this paper was presented at the Optical Fiber Communication (OFC) Conference, Los Angeles, CA, 2004.

C. Ou was with the Department of Computer Science, University of California, Davis, CA 95616 USA. He is now with AT&T Services, Inc., San Ramon, CA 94583 USA (e-mail: sam.ou@att.com).

L. H. Sahasrabuddhe is with Park, Vaughan & Fleming LLP, Davis, CA 95616 USA (e-mail: laxman@parklegal.com).

K. Zhu is with Brion Tech. Inc., Santa Clara, CA 95054 USA (e-mail: kzhu@briontech.com).

C. U. Martel and B. Mukherjee are with the Department of Computer Science, University of California, Davis, CA 95616 USA (e-mail: martel@cs.ucdavis.edu; mukherje@cs.ucdavis.edu).

Digital Object Identifier 10.1109/TNET.2005.863462

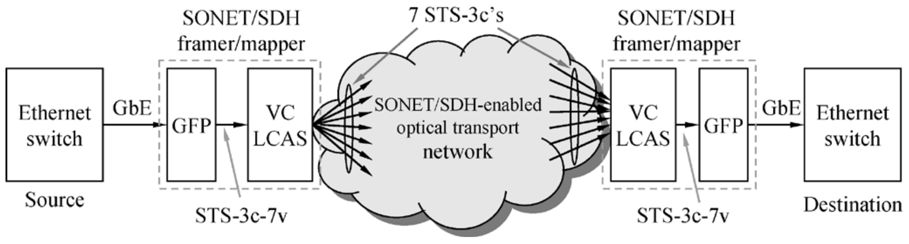


Fig. 1. Provisioning a GbE connection in a SONET/SDH-enabled optical transport network.

assumption here is that the destination node can compensate the differential delay of the VCG members via buffering. Commercial SONET/SDH framer/mapper can compensate up to ±125 ms differential delay of up to 64 VCG members via external random-access memory (RAM) [32].

Since data traffic is bursty, it is desirable that the SONET/SDH transport network can dynamically adjust the bandwidth allocated to a connection to accommodate long-term traffic fluctuation. (Short-term traffic fluctuation is more challenging to accommodate, because adjusting the bandwidth allocated to a connection typically takes multiple round-trip delays, and is not the focus of this work.)

Link-capacity-adjustment scheme (LCAS) serves this purpose. LCAS is a two-way signaling protocol built on VC, and it can dynamically adjust the bandwidth of a VC connection by adding/deleting VCG members in a hitless manner (without disrupting the traffic carried on the existing VCG members). Besides, LCAS provides some degree of resilience since it can dynamically remove failed (or add recovered) VCG members. In case of network-element failures or network congestion, LCAS can reduce the bandwidth of VC connections by removing the disrupted VCG members, *thus providing degraded services, instead of "no service" at all, as in current solutions*.

As an example, consider the provisioning of a Gigabit Ethernet (GbE) connection in an optical transport network using next-generation SONET/SDH. At the source node, the GbE connection can be mapped via GFP into STS-3c-7v, each of which is routed across the SONET/SDH network independently; at the destination node, the seven independent STS-3c's are combined into a STS-3c-7v signal and mapped back to GbE frames, as shown in Fig. 1. If a VCG member—which is a STS-3c signal—fails, the destination node can detect the failure and notify the source node to remove the failed VCG member via LCAS. When the failure gets fixed, the destination informs the source node to add the recovered VCG member back into the VCG. Before the failure is fixed, the source node can also add into the VCG another path of STS-3c free capacity if it so desires and if such a path exists.

Route computation leveraging the inverse multiplexing, or multi-path routing, capability of VC is challenging and is assumed to be the responsibility of the network-management system according to the various standards bodies. In [39], we quantified the benefits of using VC for unprotected traffic. In this work, we investigate the survivability of DoS.

B. Motivation for Survivable DoS

In a SONET/SDH-enabled optical transport network employing wavelength-division multiplexing (WDM), the failure of a network element, such as a fiber, can cause the failure of multiple VCG members, thereby leading to large data (and revenue) loss. Protection, a proactive procedure in which spare capacity is reserved during connection setup [8], [10], [14], [23]–[25], [28], is essential for recovering from such failures in a short time period. A path carrying traffic during normal operation is known as a *working* path.[1] When a working path fails, the connection is rerouted over a *backup* path. Multiple backup paths may share bandwidth if their corresponding working paths are shared-risk-link-group (SRLG) disjoint, where SRLG is an abstraction referring to a group of fibers that may be prone to a common failure [31], [33], [38].

High bandwidth efficiency and short fault-recovery time are two of the most important features of a protection scheme [15], [18], where fault-recovery time for a connection is the time duration the connection takes to properly signal/configure the nodes along the backup path before switching traffic to the backup path after a failure occurs on the working path [28].

Data networks have limited protection capabilities. For example, depending on the size of the network, current Ethernet protection schemes based on spanning-tree algorithms can take up to dozens of seconds to converge [2]. Clearly, the fault-recovery time of Ethernet is too long for voice and mission-critical applications. IP restoration is also too slow for mission-critical applications [11], [13], [30].

While SONET/SDH automatic protection switching (APS) provides protection against single-fiber failure within 50 ms, SONET/SDH APS is known to be resource inefficient. SONET APS generally incurs at least 100% (and up to 300%) backup resource overbuild [22]. (The resource overbuild of SONET APS may be reduced by applying a 1:N protection scheme. However, 1:N may not always be applicable in a given context.) Since a large portion of data traffic is inherently best effort and nonmission critical, providing 50-ms protection with a huge sacrifice in backup resources is not desirable.

In addition, protecting a DoS connection also differs from WDM protection without VC [28] and multiprotocol label switching (MPLS) tunnel restoration [19] in that one DoS connection can be inversely multiplexed onto multiple paths. With VC, multiple members of a VCG (of *one* connection) may share backup resources if these members are SRLG-disjoint; we refer to this as *intra-connection sharing*. Meanwhile, multiple members of different VCGs (of *different* connections) may share backup resources if these members are SRLG-disjoint; we refer to this as *inter-connection sharing*. Protection becomes

[1]Working path is also referred to as primary path, active path, and service path in the literature.

much more complex because a flow, consisting of multiple paths, needs to be protected.

In summary, new approaches are needed for provisioning survivable DoS connections to strike an attractive balance between resource efficiency and fault-recovery time while taking advantage of the inverse-multiplexing capability of VC.

C. Our Contribution

We propose and investigate two new approaches—Protecting Individual VCG Member (PIVM) and Provisioning fast REstorable VCG (PREV)—and the associated route-computation algorithms for dynamically provisioning survivable DoS connections. Both approaches provide 100% guarantee against single-fiber failures and degraded services against multiple-fiber failures.[2] By exploiting the inverse-multiplexing capability, our approaches maximize intra-connection sharing and exploit the tradeoff between inter-connection sharing and fault-recovery time. PIVM, which is suitable for centralized implementation, achieves high backup sharing which leads to efficient resource utilization; while PREV, which is suitable for distributed implementation, achieves fast fault recovery comparable to SONET 50-ms protection (for a typical U.S. backbone network) while still achieving modest backup sharing.

Since it may not be desirable to inversely multiplex one connection onto an arbitrary number of paths due to network control and management (NC&M) considerations and also possibly due to the restrictions imposed by the network topology, we also investigate the impact of VCG size, denoted by M, on network performance. We first prove that it is NP-complete to compute a minimum-cost VCG having two members, i.e., $M = 2$, of combined bandwidth no less than a given value. Then, we design an effective heuristic. Our results indicate that the network blocking performance for small values of M (e.g., $M = 2$ for some representative U.S. nationwide backbone network topologies) is almost as good as the case in which M is infinity.

D. Organization

The rest of this paper is organized as follows. Sections II and III present PIVM and PREV, respectively. Section IV describes our approach for controlling VCG size. Section V compares the characteristics and performance of PIVM and PREV using illustrative examples. Section VI concludes the paper.

II. Protecting Individual VCG Member (PIVM)

A. Basic Idea

PIVM is similar to shared-mesh protection, and it exploits the inverse-multiplexing capability of VC to further increase backup sharing. Denote as $\langle s, d, u\rangle$ a connection request from node s to node d requiring u units of bandwidth; and represent the capacity of path p as $B(p)$. The basic ideas of PIVM are:

[2]Single-fiber failures are the predominant form of failures in communication networks. Node failures are relatively rare compared to fiber failures because the switch fabric and switch-control unit in a carrier-class node are typically dedicated (1 + 1) protected. Therefore, nodes are assumed to be robust in this work.

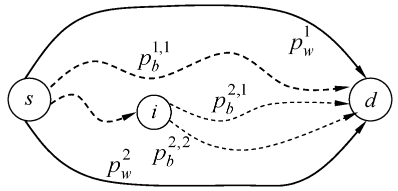


Fig. 2. Survivable DoS approach: PIVM.

1) to route connection $\langle s, d, u\rangle$ with a working VCG $\mathcal{P}_w$ of capacity u (the capacity of a VCG is the total capacity of its members);
2) to protect each working VCG member $p_w^k \in \mathcal{P}_w$ $(1 \le k \le |\mathcal{P}_w|)$ with a backup VCG $\mathcal{P}_b^k$ of capacity $B(p_w^k)$ such that any member of $\mathcal{P}_b^k$ is link-disjoint to p_w^k; and
3) to share backup resources between any two working VCG members as long as they are link-disjoint.

Please note that p_w^k and any path in $\mathcal{P}_b^l$ $(k \neq l)$ do not need to be link-disjoint. We also remark that: 1) WDM shared-mesh protection is a special case of PIVM in which $u = 1$ for every connection request; and 2) MPLS tunnel protection is a special case of PIVM in which $|\mathcal{P}_w| = 1$ and $|\mathcal{P}_b^1| = 1$ ($k = 1$ because $1 \le k \le |\mathcal{P}_w| = 1$) for every connection request.

B. An Example

To elaborate on the above ideas, consider the example in Fig. 2. The connection from node s to node d has working VCG $\mathcal{P}_w = \{p_w^1, p_w^2\}$. Working VCG member p_w^1 is protected by backup VCG $\mathcal{P}_b^1 = \{p_b^{1,1}\}$, and p_w^2 is protected by backup VCG $\mathcal{P}_b^2 = \{p_b^{2,1}, p_b^{2,2}\}$. Note that, even though VCG members $p_b^{2,1}$ and $p_b^{2,2}$ follow the same path from node s to node i, they are inversely multiplexed at node s, *not* at node i, because $p_b^{2,1}$ and $p_b^{2,2}$ are routed independently. They follow the same segment from s to i because there is enough capacity on it. Basically, $p_b^{2,1}$ and $p_b^{2,2}$ are capacity-disjoint from s to d. Backup VCG member $p_b^{1,1}$ may share backup capacity with both $p_b^{2,1}$ and $p_b^{2,2}$ if p_w^1 is link-disjoint to p_w^2 (intra-connection sharing). $p_b^{1,1}$ may share backup capacity with any member of an existing backup VCG (not shown in the figure) if p_w^1 is link-disjoint to the corresponding working VCG member (inter-connection sharing).

In case of a working VCG member failure, e.g., p_w^2, destination d detects the failure and notifies source s. Upon receiving the notification, source s: 1) removes the failed member, p_w^2, from the working VCG via LCAS; 2) signals the nodes along the backup VCG, $\mathcal{P}_b^2$, to properly configure their switches; and 3) adds all the members of the backup VCG, $\mathcal{P}_b^2$, to the working VCG via LCAS.

C. Route Computation: General Case

Before we present our route-computation approach for an incoming connection request, let us define the notations and formally state the dynamic connection-provisioning problem under PIVM constraints.

1) Notations: A network is represented as a weighted, directed graph $G = (V, E, C, \lambda)$, where V is the set of nodes, E is the set of unidirectional fibers (referred to as links), $C : E \rightarrow Z^+$ is the cost function for each link (where Z^+ denotes the set of positive integers), and $\lambda : E \rightarrow Z^+$ specifies the number of wavelengths on each link.

Every link is associated with a conflict set to identify the sharing potential between backup VCGs.[3] Let W_c be the wavelength capacity. The conflict set ν_e for link e can be defined as an integer set, $\{\nu_e^{e'} | \forall e' \in E, 0 \leq \nu_e^{e'} \leq \lambda(e') \times W_c\}$, where $\nu_e^{e'}$ represents the amount of traffic that will be rerouted on link e when link e' fails. The amount of backup capacity reserved on link e is thus $\nu_e^* = \underset{\forall e' \in E}{max}\{\nu_e^{e'}\}$. The difference $\nu_e^* - \nu_e^{e'}$ indicates the potential "free" capacity for backing up a new working VCG member which traverses link e' and whose corresponding backup VCG traverses link e.

The union of the conflict sets for all the links aggregates the per-connection-based information, and the size of the conflict set depends only on the number of links, not on the number of connections. In the absence of such a mechanism as conflict set, per-connection-based information is necessary for identifying shareable backup bandwidth [7]. It is, thus, advantageous to use conflict set since the number of connections can be significantly more than the number of links.

2) Problem Statement: We now formally state the dynamic connection-provisioning problem under PIVM constraints as follows: Given a WDM network as $G = (V, E, C, \lambda)$ and the set of existing connections (or the associated conflict set $\{\nu_e | e \in E\}$), for each incoming connection request $\langle s, d, u \rangle$, compute a working VCG $\mathcal{P}_w$ from node s to node d of bandwidth u and a set of backup VCGs (one for each working VCG member) under backup-sharing constraints while minimizing the total cost of the working and backup VCGs.

The existence version of the above problem for provisioning one connection request under the current network state is NP-complete. This is because the existence version of shared-mesh protection, which is a special case of this problem in which $u = 1$ for every connection request, has been proven to be NP-complete [27]. Therefore, practical heuristics are needed, as shown below.

3) The PIVM Heuristic: Upon the arrival of a new connection request $\langle s, d, u \rangle$, our PIVM heuristic operates as follows:

1) Compute a min-cost flow f_w from node s to node d of bandwidth u using a standard min-cost flow algorithm [5].
2) Extract as the working VCG $\mathcal{P}_w$ a set of paths constituent to flow f_w.
3) For every working VCG member $p_w^k \in \mathcal{P}_w$, compute as the backup VCG $\mathcal{P}_b^k$ an integral min-cost flow from node s to node d of bandwidth $B(p_w^k)$ while accommodating backup sharing.

The challenge here is how to take into consideration backup sharing. In WDM shared-mesh protection and MPLS tunnel restoration, we can accommodate backup sharing by simply manipulating link cost when computing a backup path. For example, if link e has a shareable wavelength, then the cost of link e can be redefined much smaller than the original cost of link e when computing the backup path [6], [8], [34], [36]. Manipulating link cost does not apply here because the cost of a link depends on the amount of flow traversing the link, and we do not know in advance the amount of flow on every link.

We accommodate backup sharing by introducing parallel links. For a given working VCG member p_w^k, calculate for every link e the amount of shareable bandwidth, $B_z(e)$, as follows:

$$B_z(e) = \underset{\forall e' \in E}{max}\left\{\nu_e^{e'}\right\} - \underset{\forall e' \in p_w^k}{max}\left\{\nu_e^{e'}\right\}. \qquad (1)$$

For every link e whose $B_z(e) > 0$, we introduce a link e' parallel to e, i.e., links e and e' originate from the same upstream node and terminate at the same downstream node. The bandwidth of link e' will be $B_z(e)$. The cost of link e' will be far smaller than that of link e. When a min-cost flow algorithm is applied to a graph so modified, the algorithm will always prefer link e' to link e as long as link e' has available bandwidth because link e' has much less cost. Therefore, minimizing the cost of the flow is equivalent to maximizing backup sharing in this case.

A formal specification of PIVM is shown in Algorithm 1. The computational complexity of Algorithm 1 depends on the min-cost flow algorithm. If we employ the enhanced capacity-scaling algorithm, which has complexity $O((|E| \log |V|)(|E| + |V| \log |V|))$ [5], then the complexity of Algorithm 1 will be $O((u \cdot |E| \log |V|)(|E| + |V| \log |V|))$. In particular, the complexity of Step 1 is $O((|E| \log |V|)(|E|+|V| \log |V|))$, the complexity of Step 2 is $O(u \cdot |V|^2)$, the complexity of Step 3 is $O((u \cdot |E| \log |V|)(|E| + |V| \log |V|))$, and the complexity of Step 4 is $O(1)$.

Algorithm 1 PIVM

Input: $\langle s, d, u \rangle$, $G = (V, E, C, \lambda)$, $B_a : E \rightarrow Z^+$ specifying the amount of available bandwidth on every link, and the conflict set $\{\nu_e | e \in E\}$.

Output: A minimum-cost working VCG $\mathcal{P}_w$ and a set of backup VCGs (one for each working VCG member); otherwise NULL if no eligible solution is found.

```
1) Compute in $G$ an integral min-cost flow $f_w$ of bandwidth $u$ from node $s$ to node $d$ based on the available bandwidth of each link; return NULL if $f_w$ is not found.
2) Extract as the working VCG $\mathcal{P}_w$ a set of paths constituent to flow $f_w$ and update $B_a$.
3) For each working VCG member $p_w^k \in \mathcal{P}_w$: Calculate for every link $e$ the amount of shareable bandwidth, $B_z(e)$, according to (1). Compute an integral min-cost flow $f_b^k$ of bandwidth $B(p_w^k)$ from node $s$ to node $d$ in an auxiliary graph $G' = (V, E', C', \lambda)$ with available bandwidth function $B'_a : E' \rightarrow Z^+$, where
    a) $\forall e \in E \wedge B_a(e) > 0 \wedge e \notin p_w^i$, $e \in E'$, $B'_a(e) = B_a(e)$, and $C'(e) = |V| \cdot C(e)$
    b) $\forall e \in E \wedge B_z(e) > 0 \wedge e \notin p_w^i$, $e \in E'$, $B'_a(e) = B_z(e)$, and $C'(e) = 1$ (If link $e$ has already been
```

[3]A wavelength-routed WDM network can be wavelength continuous, in which a connection is required to occupy the same wavelength throughout its path in the network, or it can be wavelength convertible, in which a connection may use different wavelengths on different links due to the existence of wavelength-conversion devices [26]. In this work, we assume that every node is wavelength convertible because wavelength conversion comes for free due to the optical-electrical-optical (OEO) conversion at every node in current communication networks. In the wavelength-continuous case, we would associate a conflict set to a wavelength. The conflict set defined here is related to the conflict vector in [25], the aggregated square matrix in [21], and the "bucket" link metric in [35].

added to E' in Step 3a, this one will be parallel to the existing one; in other words, there can be two links between the same node pair in G').

Return NULL and undo any of the changes if f_b^k is not found; otherwise, let $\mathcal{P}_b^k$ be a set of paths constituent to the flow f_b^k (note that if two parallel links both appear in f_b^k, they should be combined in $\mathcal{P}_b^i$) and update B_a and ν_e.

4) Return $\mathcal{P}_w$ as the working VCG and $\{\mathcal{P}_b^k\}$ as the set of backup VCGs.

We remark that PIVM is suitable for centralized implementation because: 1) PIVM needs the detailed routing information of all the existing connections, or the conflict set of all the links, to measure the backup-sharing potential for provisioning a new connection request in Step 3 of Algorithm 1; and 2) after a connection is provisioned, the conflict set on any link that is traversed by the backup VCGs of the connection needs to be updated. In a centralized network-management system, the detailed information of all the existing connections is available, and updating the conflict set on some links is straightforward. However, in a distributed network-management system, the detailed routing information of all the existing connections may not be available at every node. Furthermore, propagating conflict-set updates of all the links traversed by the backup VCGs may incur a large signaling volume, and maintaining a consistent view of the entire network among all the nodes may become difficult. In the following section, we propose an alternate approach suitable for distributed implementation.

III. Provisioning fast REstorable VCG (PREV)

A. Basic Idea

PREV aims at fast protection switching by judiciously sharing backup capacity such that, when a working VCG member fails, no backup configuration is needed at intermediate nodes along the backup path (but, of course, backup configuration at the source and destination nodes is necessary). As shown in [16], to avoid backup configuration, the following constraints apply: 1) only connections having the same source-destination pair can share backup bandwidth; and 2) a backup VCG member can only be shared end-to-end. Under these constraints, PREV works as follows: 1) pre-select a backup path, p_{sd}^b, for each node pair $\langle s, d\rangle$; and 2) route each connection request such that both inter-connection sharing (with existing connections from s to d) and intra-connection sharing (by inverse multiplexing) can be maximized.

B. An Example

To elaborate on the above ideas, consider the example in Fig. 3. Suppose that paths p_{sd}^1, p_{sd}^2, and p_{sd}^3 are link-disjoint to path p_{sd}^b and there is sufficient bandwidth available on these paths (for the purpose of illustration). Assume that path p_{sd}^1 is link-disjoint to paths p_{sd}^2 and p_{sd}^3. Let p_{sd}^b be the backup path for any connections from node s to node d under PREV. To provision a GbE connection from node s to node d, we can route STS-1-11v bandwidth on p_{sd}^1 and STS-1-10v bandwidth

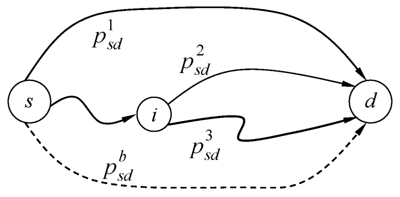


Fig. 3. Survivable DoS approach: PREV.

on path p_{sd}^2 (or STS-1-10v bandwidth on path p_{sd}^3, or STS-1-xv bandwidth on p_{sd}^2 and STS-1-yv bandwidth on p_{sd}^3 subject to $x + y = 10$). The amount of backup bandwidth that needs to be reserved on p_{sd}^b is only STS-1-11v since p_{sd}^1 is link-disjoint to paths p_{sd}^2 and p_{sd}^3. As a result, we can protect a GbE connection with only STS-1-11v bandwidth (about 600 Mb/s) via intra-connection sharing. In general, if the number of mutually link-disjoint paths between node s and node d is larger, then the amount of backup bandwidth that needs to be reserved can be further reduced.

Later, suppose a new connection request from node s to node d requiring bandwidth STS-1 arrives before the GbE connection leaves. The new connection can be routed on path p_{sd}^2 (or p_{sd}^3) (assuming they have free bandwidth of STS-1 or higher) without reserving additional backup bandwidth on path p_{sd}^b due to inter-connection sharing.

In case of a working VCG member failure, e.g., p_{sd}^1, destination node d notifies source s via LCAS. Upon receiving the notification, source s removes the failed member (p_{sd}^1) via LCAS, configures itself to use path p_{sd}^b, and adds to the working VCG path p_{sd}^b with capacity equal to the amount of capacity previously routed on the failed working VCG member (p_{sd}^1) via LCAS. Note that there is no backup configuration at the intermediate nodes along the backup path p_{sd}^b after a failure occurs.

C. Pre-Select a Backup Path for Every Node Pair

Under PREV, a proper backup path needs to be chosen for every node pair in advance. One reasonable criterion for pre-selecting a backup path for each node pair is load balancing, i.e., to select a path p_{sd}^b for every node pair $\langle s, d\rangle$ such that the maximum number of times a link is traversed by all the backup paths $\{p_{sd}^b | \forall s \neq d \in V\}$ is minimized. This problem, referred to as balanced backup selection, is NP-complete for a directed graph because: 1) the disjoint-path problem stated below is NP-complete for any $k \geq 2$ [9], [29]; and 2) the disjoint-path problem with $k = |V| \cdot (|V| - 1)$ is a special case of the balanced backup-selection problem with the maximum number of times a link is traversed by all the backup paths being one.

Disjoint-Path Instance: A directed graph $G = (V, E)$ and distinct node pairs $\langle s_1, d_1\rangle, \cdots, \langle s_k, d_k\rangle$ ($s_i \neq d_i \in V$).

Disjoint-Path Question: Do there exist mutually link-disjoint paths, $p_1, \cdots, p_k$, in G such that p_i joins s_i and d_i ($1 \leq i \leq k$)?

Besides the load-balancing concern, a poorly chosen backup path, p_{sd}^b, can significantly reduce the amount of working flow which is link-disjoint to p_{sd}^b. For example, Fig. 4 demonstrates that a poorly chosen backup path, $\langle s, i, j, d\rangle$, disconnects the source node s from the destination node d even if sufficient capacity is available on every link because the working flow needs to be link-disjoint to path $\langle s, i, j, d\rangle$. If an appropriate backup

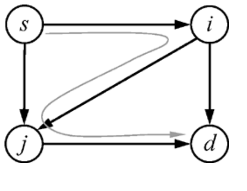


Fig. 4. Inappropriate backup path $\langle s, i, j, d\rangle$.

path, e.g., $\langle s, i, d\rangle$, is chosen, some link-disjoint flow can be carried on path $\langle s, j, d\rangle$. Clearly, proper approaches are needed to avoid this type of trap situation [27], [37].

Denote as $MinCut(S, D)$ a minimum cut with respect to capacity between node s and node d in $G = (V, E, C, \lambda)$ representing the given network, where $S \cup D = V$, $S \cap D = \phi$, $s \in S$, and $d \in D$ (i.e., we consider two nonoverlapping partitions of the network, one partition containing the source node s and the other containing the destination node d). A "good" backup path should have the property of traversing only one link per $MinCut(S, D)$ for any $MinCut(S, D)$ to maximize the amount of working flow. (If a path traverses more than one link of a minimum-cut $MinCut(S, D)$, then the amount of flow disjoint to the path will be unnecessarily reduced at $MinCut(S, D)$ due to the equivalence of maximum flow and minimum cut.)

Let $\mathcal{P}_{sd}$ be a set of mutually link-disjoint paths between node s and node d with maximum cardinality. Clearly, any path in $\mathcal{P}_{sd}$ only traverses one link per $MinCut(S, D)$. Load-balancing criterion can then be employed to decide which path in $\mathcal{P}_{sd}$ should be selected as p^b_{sd}.

In summary, the backup path for every node pair is computed as follows:

1) For every node pair $\langle s, d\rangle$ $(s \neq d \in V)$, compute $\mathcal{P}_{sd}$ by applying a maximum-flow algorithm to graph G with unit link capacity.
2) Randomly choose one path from $\mathcal{P}_{sd}$ as p^b_{sd} for every node pair $\langle s, d\rangle$.
3) Let m be the maximum number of times any link is traversed by the set of paths $\{p^b_{sd}\}$. Let L_k be the set of links traversed by k paths in $\{p^b_{sd}\}$, $1 \leq k \leq m$. Find a node pair $\langle s, d\rangle$ such that: (i) p^b_{sd} traverses the most links in L_m; and (ii) there exists at least one path $p \in \mathcal{P}_{sd}$ such that p does not traverse any link in L_m and L_{m-1}. Terminate if no such node pair is found; otherwise, replace p^b_{sd} by p and repeat this step.

We remark that the above procedure is guaranteed to terminate because one iteration of Step 3 reduces the size of L_m by at lease one, $|L_m| \leq |E|$, $m \leq |V| \cdot (|V| - 1)$, and m does not increase its value. We also remark that the above procedure is best suited for uniform traffic. For nonuniform traffic, we can proportionally weight each backup path p^b_{sd} according to the amount of traffic from node s to node d and make appropriate adjustments in Step 3 for load balancing.

The complexity of the above procedure is $O(|V|^5)$. In particular, the complexity of Step 1 is $O(|V|^5)$, the complexity of Step 2 is $O(|V|^2)$, and the complexity of Step 3 is $O(|V|^5)$. Since this procedure only needs to be run once for every topology, the high complexity may not be a problem.

D. Route Computation

Every backup path, as computed in Section III-C, is associated with a conflict set to identify backup-sharing potential. The conflict set ν_{sd} for backup path p^b_{sd} can be represented as an integer set, $\{\nu^e_{sd} | \forall e \in E\}$, where ν^e_{sd} represents the amount of traffic that will be rerouted on backup path p^b_{sd} when link e fails. The amount of backup capacity reserved on backup path p^b_{sd} is thus $\nu^*_{sd} = \underset{\forall e \in E}{max}\{\nu^e_{sd}\}$.

Upon the arrival of a new connection request $\langle s, d, u\rangle$, our PREV routing algorithm computes a working flow f, which is link-disjoint to the backup path p^b_{sd}, from node s to node d of bandwidth u to jointly minimize the cost of the working flow and the incremental cost on the backup path. Due to backup sharing, $(\nu^*_{sd} - \nu^e_{sd})$ units of working flow can be routed on link e without increasing the amount of backup capacity reserved on the backup path p^b_{sd}; additional amount of working flow routed on link e will cause additional amount of backup capacity to be reserved. The basic idea of our PREV routing algorithm is to route the working flow in a way such that the cost of the working flow and the incremental cost on the backup path are jointly minimized. Below, we formulate the PREV routing problem mathematically and transform the formulation to solve it efficiently.

1) Problem Formulation: Represent the network as a weighted, directed graph $G = (V, E, C, \lambda)$ as before. Let $B_a : E \rightarrow Z$ specify the available bandwidth on every link. Denote as C^b_{sd} the cost of the backup path p^b_{sd} and as U^b_{sd} the amount of backup bandwidth reserved on p^b_{sd} before we process the new connection request. Let $B(p^b_{sd})$ be the maximum available bandwidth along path p^b_{sd}, i.e., $B(p^b_{sd}) = \underset{\forall e \in p^b_{sd}}{min}\{B_a(e)\}$.

Suppose that the working flow, $f : E \rightarrow Z$, is decided for the new connection request. Then, the amount of additional bandwidth to be reserved on the backup path p^b_{sd}, denoted as U_Δ, can be calculated as follows: $U_\Delta = (\underset{\forall e \in E}{max}\{\nu^e_{sd} + f(e)\} - U^b_{sd})$. Therefore, the cost of admitting this new connection request $\langle s, d, u\rangle$ is $(\sum_{\forall e \in E} f(e) \times C(e) + U_\Delta \times C^b_{sd})$.

Since our objective is to jointly minimize the cost $(\sum_{\forall e \in E} f(e) \times C(e) + U_\Delta \times C^b_{sd})$, the working flow f can not be decided without considering the incremental cost on the backup path. However, note that U_Δ is bounded by $min\{B(p^b_{sd}), u\})$. We can try U_Δ for every value in $[0, min\{B(p^b_{sd}), u\}]$ to jointly minimize $(\sum_{\forall e \in E} f(e) \times C(e) + U_\Delta \times C^b_{sd})$.

The PREV routing problem can be mathematically formulated as follows. The formulation turns out to be an integer linear program (ILP). In the formulation, $B_z : E \rightarrow Z$ is a function specifying, for every link, the amount of bandwidth which can be used by the working flow and which will not cause additional bandwidth to be reserved on the backup path p^b_{sd}, i.e.,

$$B_z(e) = min\left\{B_a(e), U^b_{sd} - \nu^e_{sd}\right\}, \quad \forall e \in E \quad (2)$$

Objective :

$$\text{Minimize} \quad \left(\sum_{\forall ij \in E} f(ij) \times C(ij)\right) + U_\Delta \times C^b_{sd} \quad (3)$$

Constraints :

$$\sum_{\forall j, ij \in E} f(ij) - \sum_{\forall j, ji \in E} f(ji) = \begin{cases} u, & \text{if } i = s \\ -u, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$f(ij) \le min\{B_a(ij), u\} \quad \forall ij \in E \tag{5}$$
$$f(ij) \le B_z(ij) + U_\Delta \quad \forall ij \in E \tag{6}$$
$$0 \le U_\Delta \le min\{B(p_{sd}^b), u\} \tag{7}$$
$$f(ij), U_\Delta : int. \tag{8}$$

Our objective function (3) jointly minimizes the cost of the working flow and the incremental cost on the backup path. Equation (4) accounts for the flow-conservation constraints for the working flow. Equation (5) states that the working flow can utilize no more than the amount of capacity available on any link and the amount of capacity needed by the connection request. Equation (6) ensures that the working flow will not cause more than U_Δ units of additional backup bandwidth to be reserved. Equation (7) constrains the range of the variable U_Δ.

One common approach for solving an ILP is to use a commercial solver such as CPLEX. Such an approach typically takes a long time and may not scale well given today's limited computational power. As a result, we resort to a different approach in which we only examine the crucial points in the search space without losing the optimality of the solution.

2) The PREV Algorithm: Our first observation is that the above ILP reduces to a standard min-cost flow problem if U_Δ is a constant. If U_Δ is a constant, (5) and (6) can be transformed to constrain the available link capacity with a new function $B_a : E \to Z$, where

$$B'_a(e) = min\{B_a(e), u, B_z(e) + U_\Delta\}. \tag{9}$$

The only constraint left in the ILP is (4), which is exactly the specification of the min-cost flow problem [5]. Therefore, if U_Δ is a constant, the ILP can be solved by applying a min-cost flow algorithm to graph G with the new available link-capacity function B'_a defined in (9).

Based on this observation, a straightforward approach could be to apply a min-cost flow algorithm for each value of $U_\Delta \in [0, min\{B(p_{sd}^b), u\}]$ and find the U_Δ corresponding to the minimum value of the objective $(\sum_{\forall e \in E} f(e) \times C(e) + U_\Delta \times C_{sd}^b)$.

A more efficient approach is to apply a modified binary search on U_Δ, as discussed below. Decompose the objective function into two terms: let $c_w(U_\Delta) = \sum_{\forall e \in E} f(e) \times C(e)$ and $c_b(U_\Delta) = U_\Delta \times C_{sd}^b$ (please note that, even though there is no U_Δ in the definition of c_w, the value of U_Δ affects the working flow f, which in turn affects c_w). Clearly, $c_b(U_\Delta)$ is a linear function of U_Δ. We show that $c_w(U_\Delta)$ has the property shown in Theorem 1.

Theorem 1:

$$c_w(U_\Delta^1) - c_w(U_\Delta^1 + 1) \ge c_w(U_\Delta^2) - c_w(U_\Delta^2 + 1) \quad \forall\, 0 \le U_\Delta^1 < U_\Delta^2 < min\{B(p_{sd}^b), u\}. \tag{10}$$

Proof: Please refer to Appendix I. ■

This property provides the basis for the modified binary-search procedure. Let U_Δ^* $(0 \le U_\Delta^* \le min\{B(p_{sd}^b), u\})$ be the optimal value of U_Δ leading to the minimum of $c_w(U_\Delta) + c_b(U_\Delta)$. For an arbitrary U_Δ $(0 \le U_\Delta < min\{B(p_{sd}^b), u\})$, U_Δ^* can be decided as follows.

1) If $c_w(U_\Delta) - c_w(U_\Delta + 1) < C_{sd}^b$, then $U_\Delta^* \in [0, U_\Delta]$.
2) If $c_w(U_\Delta) - c_w(U_\Delta + 1) = C_{sd}^b$, then $U_\Delta^* = U_\Delta$.
3) Otherwise, $U_\Delta^* \in [U_\Delta, min\{B(p_{sd}^b), u\}]$.

The property of c_w, as shown in (10), ensures the correctness of the binary search. A formal specification of our PREV algorithm is shown in Algorithm 2.

Algorithm 2 PREV

```
Input: $\langle s, d, u\rangle$, $G = (V, E, C, \lambda)$, and $B_a$.
Output: A working VCG $\mathcal{P}_w$ and the amount of additional bandwidth $U_\Delta$ to be reserved on $p_{sd}^b$; otherwise NULL if no eligible solution is found.
1) Return NULL if no flow of capacity $u$ can be found.
2) Calculate $B_z$ according to (2).
3) $U_\Delta^L \leftarrow 0$; $U_\Delta^H \leftarrow min\{B(p_{sd}^b), u\}$.
4) Repeat until $U_\Delta^L = U_\Delta^H$
   a) $U_\Delta^M \leftarrow (U_\Delta^L + U_\Delta^H)/2$
   b) If $c_w(U_\Delta^M) - c_w(U_\Delta^M + 1) < C_{sd}^b$, then $U_\Delta^H \leftarrow U_\Delta^M$;
      if $c_w(U_\Delta^M) - c_w(U_\Delta^M + 1) = C_{sd}^b$, then $U_\Delta^L \leftarrow U_\Delta^M$
      and $U_\Delta^H \leftarrow U_\Delta^M$; otherwise $U_\Delta^L \leftarrow U_\Delta^M$ ($c_w(U_\Delta^M)$
      and $c_w(U_\Delta^M + 1)$ are computed by applying a min-cost flow algorithm to $G$ with available link capacity redefined in (9)).
5) Let $f_w$ be a min-cost flow corresponding to $U_\Delta^L$ and
   $\mathcal{P}_w$ be a set of paths constituent to $f_w$. Return $\mathcal{P}_w$ and
   $U_\Delta = U_\Delta^L$.
```

The complexity of Algorithm 2 depends on the min-cost flow algorithm. If we employ the enhanced capacity-scaling algorithm, which has complexity $O((|E| \log |V|)(|E| + |V| \log |V|))$ [5], then the complexity of Algorithm 2 will be $O((\log u) \cdot (|E| \log |V|)(|E| + |V| \log |V|))$. In particular, the complexity of Step 1 is $O((|E| \log |V|)(|E| + |V| \log |V|))$; the complexity of Step 2 is $O(|E|)$; the complexity of Step 3 is $O(1)$; the complexity of Step 4 is $O((\log u) \cdot (|E| \log |V|)(|E| + |V| \log |V|))$; and the complexity of Step 5 is $O(u|V|^2)$.

In contrast to PIVM, PREV is very suitable for distributed implementation. To measure the backup-sharing potential for a new connection request, PREV only needs the conflict set of the backup path, which can be easily maintained at the source node. After a connection request is provisioned, the source node updates the conflict set of the backup path and there is no need to propagate the update. All the information which needs to be propagated is just the remaining capacity of every link that is traversed by either the working VCG or the backup path of the connection.

IV. Route Computation With Extensions to Control the Number of VCG Members

Both Algorithms 1 and 2 use a min-cost flow algorithm to compute VCGs. One limitation of the min-cost flow algorithms is that the resultant flow can have an arbitrary number of paths. In practice, we may want to limit the number of VCG members to reduce NC&M overhead, e.g., requiring that every connection request $\langle s, d, u\rangle$ can not be inversely multiplexed into more than M paths. This additional constraint significantly increases the complexity of the problem. Below, we show that it is NP-complete to compute $M = 2$ paths of combined bandwidth u and minimum cost.

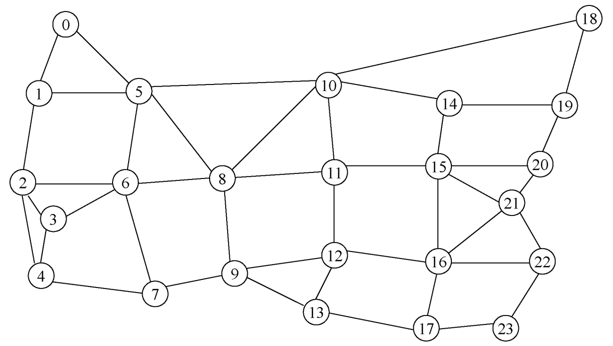


Fig. 5. Sample U.S. nationwide topology used in this work.

Instance: A graph $G = (V, E, C, \lambda)$, available bandwidth function $B_a : E \rightarrow Z$, a connection request $\langle s, d, u \rangle$, and a positive integer w.

Question: Are there two paths from s to d of total bandwidth no less than u and total cost no more than w in G?

Theorem 2: The above problem, referred to as Capacitated Minimum-Cost-Multi-Path (CMCMP) problem, is NP-complete.

Proof: Please refer to Appendix II. ■

Therefore, it is NP-complete to compute up to $M (M \geq 2)$ paths of combined bandwidth u and minimum cost. We modify a min-cost flow algorithm to compute up to M paths of combined bandwidth u and low cost (may not be minimum cost). Our modification is based on the observation that, if the capacity of any path constituent to flow f (of capacity u) is at least $\lceil u/M \rceil$, then flow f can have no more than M paths.

Given a graph $G = (V, E, C, \lambda)$, available bandwidth function B_a, a connection request $\langle s, d, u \rangle$, and positive integer M, our modified min-cost flow algorithm works as follows:

1) Redefine the available bandwidth for any link $e \in E$ as $B_a^*(e) = \lfloor B_a(e)/\lceil u/M \rceil \rfloor$.
2) Apply a min-cost flow algorithm on G with available bandwidth function B_a^* to compute flow f from node s to node d of capacity $min\{u, m\}$.
3) Return failure if f is not found. Otherwise, iteratively apply a shortest-path algorithm on f to extract a set of paths $\mathcal{P}$ constituent to f. Clearly, $|\mathcal{P}| \leq M$.
4) If $u \geq M$, then: scale the capacity of any path in $\mathcal{P}$ by a factor of $\lceil u/M \rceil$; let p be the last path extracted by the shortest-path algorithm; and reduce the capacity of p by $(\lceil u/M \rceil \cdot M - u)$.
5) Return $\mathcal{P}$.

The complexity of the above algorithm is the complexity of the min-cost flow algorithm plus $O(|E| + M \cdot |V|^2)$. Since min-cost flow algorithms typically have much higher complexity than $O(|E| + M \cdot |V|^2)$ [5], the complexity of the above algorithm is basically the same as the complexity of the min-cost flow algorithm.

V. Performance: PIVM Versus PREV

We simulate a dynamic network environment with the assumptions that the connection-arrival processes are independent of one another at all nodes and the connection-holding time follows a negative exponential distribution. The capacity of each wavelength is OC-192 (approximately 10 Gb/s), as per today's practical channel speeds. The number of connection requests follows the bandwidth (bps) distribution $50\ \text{M} : 100\ \text{M} : 150\ \text{M} : 600\ \text{M} : 1\ \text{G} : 2.5\ \text{G} : 5\ \text{G} : 10\ \text{G} = 100 : 50 : 20 : 10 : 10 : 4 : 2 : 1$ (which is close to the bandwidth distribution in a practical network). Connection requests are uniformly distributed among all node pairs. An example U.S. nationwide network topology with 16 wavelengths per fiber is shown in Fig. 5. Load (in Erlang) is defined as connection-arrival rate times average holding time times a connection's average bandwidth normalized in the unit of OC-192 (10 Gb/s).

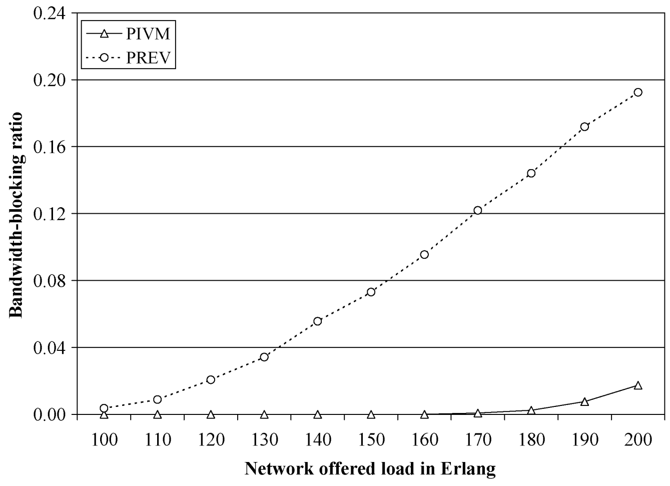


Fig. 6. Bandwidth-blocking ratio.

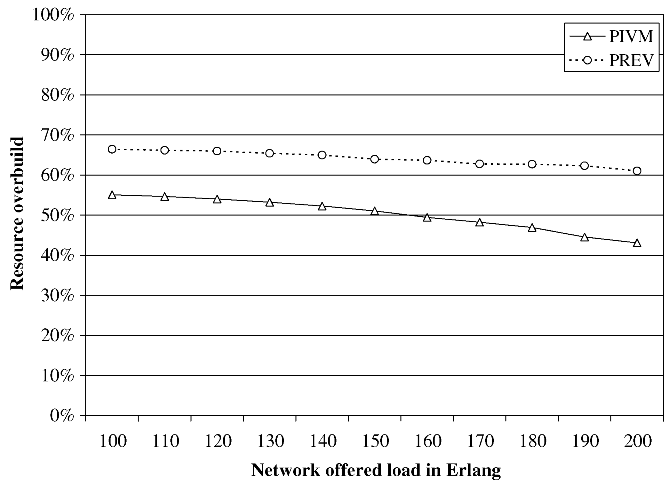


Fig. 7. Resource overbuild.

A. Bandwidth-Blocking Ratio

Bandwidth-blocking ratio is defined as the amount of bandwidth blocked over the amount of bandwidth offered. Please note that pure blocking probability, defined as the percentage of the *number* of connections blocked, cannot reflect the effectiveness of the algorithm as connections have different bandwidth requirements. Fig. 6 shows that PIVM has much lower bandwidth-blocking ratio than PREV does. This is because PREV constrains backup sharing to achieve faster fault recovery. Therefore, PIVM has much more flexibility in backup sharing, e.g., different connections between different node pairs can

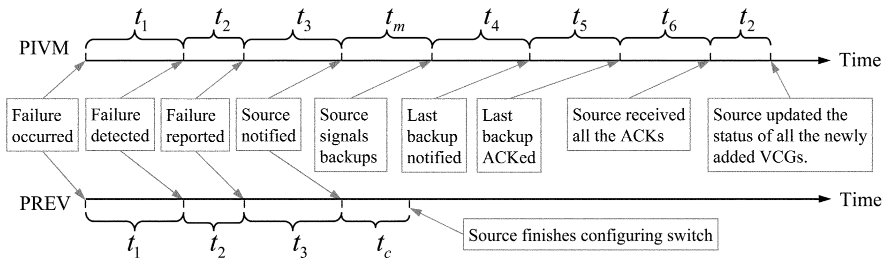

PIVM
t_1 t_2 t_3 t_m t_4 t_5 t_6 t_2
Time
Failure occurred
Failure detected
Failure reported
Source notified
Source signals backups
Last backup notified
Last backup ACKed
Source received all the ACKs
Source updated the status of all the newly added VCGs.
PREV
Time
t_1 t_2 t_3 t_c
Source finishes configuring switch

Fig. 8. Illustration of fault-recovery time based on LCAS.

share backup bandwidth under PIVM but not PREV, which leads to reduced bandwidth-blocking ratio.

B. Resource Overbuild

To quantify the amount of extra resources needed for providing protection as the percentage of the amount of resources required without protection, we employ a performance metric called resource overbuild [20]. *Resource overbuild* is defined as the amount of backup capacity (weighted by the average hop distance of backup VCG members) over the amount of working capacity (weighted by the average hop distance of working VCG members). Fig. 7 shows that both PREV and PIVM have much lower resource overbuild, about 60%–70% for PREV and 40%–60% for PIVM, than SONET/SDH's 100+% protection overhead. The resource overbuild of PIVM is lower than that of PREV because PREV applies constraints on backup sharing to achieve faster fault recovery.

C. Fault-Recovery Time

The reduced bandwidth-blocking ratio and increased backup sharing for PIVM comes for a price, namely, longer fault-recovery time. *Fault-recovery time* is defined as the time duration from the instant a failure occurs to the instant all the disrupted working VCG members have been successfully rerouted to their backup VCG members. The longest fault-recovery time is the maximal fault-recovery time among any possible single-link failure scenarios. Let t_p be the propagation delay of the longest path in the network, t_c be the switch-configuration time, and t_m be the message-processing time (including queueing time). The fault-recovery time for PIVM and PREV can be calculated as follows and is illustrated in Fig. 8.

1) PIVM: After a working VCG member fails, the destination node detects the failure in time t_1 (generally $t_1 \leq t_p$). The destination node reports the failure by changing the status of the failed working VCG member to FAIL. Because LCAS reports VCG member status periodically, this step may take a period of time t_2. The status of the failed member reaches the source node after time t_3 $(t_3 \leq t_p)$. The source node then removes the failed VCG member, and it generates and sends a message to all the nodes along the backup VCG to be activated. This step takes mainly the message-processing time, t_m. The source node also starts configuring its switch. After t_4 $(t_4 \leq t_p)$, all the nodes along the backup VCG receives the message from the source node. All the nodes along the backup VCG process this message, configure their switches in time t_5, $t_5 = t_m + t_c$, and then signal to the source node that their switches are properly configured. The source node receives the acknowledgment from all the nodes along the backup VCG in time t_6 $(t_6 \leq t_p)$. The source node then changes to ADD the status of all the members in the backup VCG in time t_2. At this point, if the switch at the source node has finished configuring itself, the source node can transmit payload on those newly added VCG members; otherwise, the source node waits until the switch configuration is finished before it starts to transmit the payload.

In summary, the total fault-recovery time for PIVM is

$$\begin{aligned} T_{\mathrm{PIVM}} &= t_1 + t_2 + t_3 + t_m + max\{t_c, t_4 + t_5 + t_6 + t_2\} \\ &= t_1 + t_2 + t_3 + t_m + t_4 + (t_m + t_c) + t_6 + t_2 \\ &\leq t_p + t_2 + t_p + t_m + t_p + (t_m + t_c) + t_p + t_2 \\ &= 4 \cdot t_p + 2 \cdot t_2 + 2 \cdot t_m + t_c \end{aligned}$$

The value of t_2 can be decided as follows. LCAS reports the status of eight VCG members in a 2-ms cycle [3], [4]. As we will show in Section V-D, small VCG sizes (no more than eight) can achieve almost as good a performance as large VCG sizes for typical U.S. backbone networks, as in Fig. 5. As a result, LCAS can report the status of all the VCG members in one cycle for our purpose, and $t_2 = 2$ ms. For typical U.S. backbone networks, $t_p \simeq 25$ ms. The configuration time of current switches is typically around 5 ms [28]. The message-processing time plus queueing delay, t_m, may fluctuate significantly. The value of t_m may be a few milliseconds if an OXC is equipped with powerful processors and may be on the order of hundred milliseconds if an OXC is equipped with less powerful processors, which might be the case for some of the commercially available OXCs. Assuming $t_m = 50$ ms, the longest fault-recovery time for PIVM is about 209 ms for a typical U.S. backbone network. Clearly, PIVM has much faster fault recovery compared to data networks such as Ethernet, whose fault recovery is on the order of seconds [2].

2) PREV: Compared to PIVM, PREV has much faster fault recovery. The main reason is that, unlike PIVM which introduces on top of LCAS additional signaling which leads to multiple round-trip delay and time-consuming message queueing and processing, PREV utilizes only the standard LCAS operations, which employ SONET/SDH overhead bytes. As a result, PREV does not require any message queueing/processing time other than the LCAS operation cycle. In addition, for PREV, there is no need to configure switches along the backup path except at the source node and the destination node.

The fault-recovery time for PREV can be calculated as follows. After a working VCG member fails, the destination node

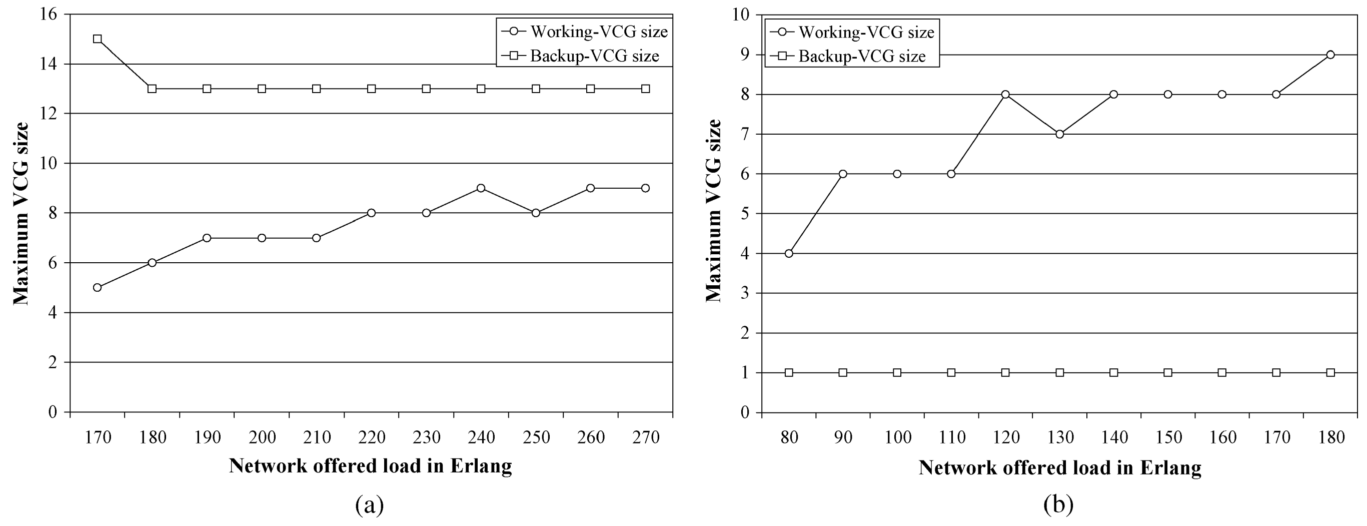


Fig. 9. Maximum VCG size ($M = +\infty$ for both working and backup VCGs). (a) PIVM. (b) PREV.

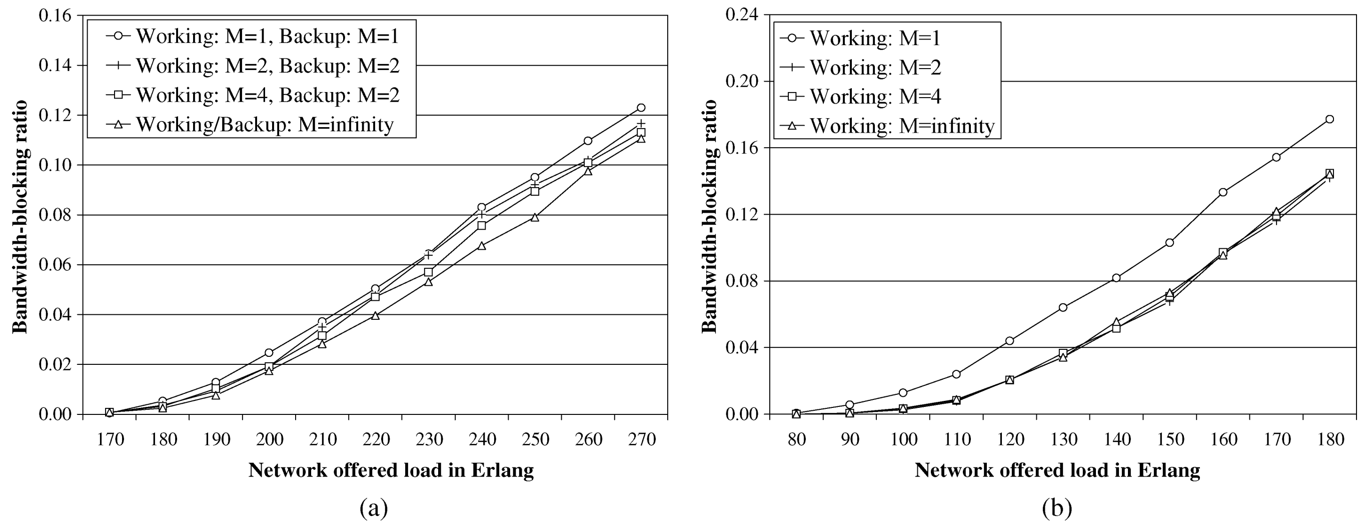


Fig. 10. Impact of VCG size on bandwidth-blocking ratio. (a) PIVM. (b) PREV.

detects the failure in time t_1 ($t_1 \leq t_p$). The destination node reports the failure in time t_2 by changing the status of the failed working VCG member to FAIL. The destination node then configures its switch. The status of the failed member reaches the source node after time t_3 ($t_3 \leq t_p$). The source node then removes the failed VCG member, changes to ADD the status of the backup path with appropriate capacity, and configures its switch. Once the switch configuration is done at the source node, the source node can start to transmit payload on the backup path. In summary, the total fault-recovery time for PREV is

$$\begin{aligned} T_{\text{PREV}} &= t_1 + t_2 + t_3 + t_c \\ &\leq t_p + t_2 + t_p + t_c \end{aligned}$$

Therefore, under PREV, the longest fault-recovery time for a typical U.S. backbone network is 57 ms ($t_p \simeq 25$ ms, $t_2 = 2$ ms, and $t_c = 5$ ms), which is comparable to SONET 50-ms protection.

Please note that we consider the longest fault-recovery time above. Since a typical connection traverses less than half of the network diameter and signal-propagation delay is the dominant component of fault-recovery time, the average fault-recovery time will be much shorter than the longest fault-recovery time.

It can also be observed from the above discussion that PREV has very little signaling overhead after a failure occurs. As a result, PREV requires less control bandwidth compared to PIVM.

D. Impact of VCG Size

So far, there was no constraint on VCG size. Fig. 9 shows that both the working-VCG size and the backup-VCG size for PIVM and the working-VCG size for PREV can be quite large. While large VCG size may improve network performance in terms of bandwidth-blocking ratio, large VCG size may not be desirable in practice because it increases implementation complexity and NC&M overhead. A natural question is: what is the "optimal" VCG size to strike a good balance between performance and complexity?

Fig. 10, which resulted from our heuristic on controlling the number of VCG members described in Section IV, shows that small VCG sizes can achieve almost as good performance as large VCG sizes. As shown in Fig. 10(a), for PIVM, the bandwidth-blocking ratio for the case in which working VCGs can be inversely multiplexed on up to four paths and backup VCGs can be inversely multiplexed on up to two paths is close to the case in which both working and backup VCGs can be inversely multiplexed on an unrestricted number of paths. For PREV, as shown in Fig. 10(b), the bandwidth-blocking ratio for the case in which working VCGs can be inversely multiplexed on up to two paths is almost the same as the case in which working VCGs can be inversely multiplexed on an unrestricted number of paths. The reason that $M = 2$ is a good choice for PREV is that the average number of mutually link-disjoint paths for the topology shown in Fig. 5 is about three. Since the backup path for every node pair is chosen from a set of mutually link-disjoint paths of maximum cardinality, the remaining two mutually link-disjoint paths, which are link-disjoint to the backup path, will be chosen to carry the working traffic because this setting leads to maximum backup sharing (please refer to Algorithm 2). The fundamental law here is the maximum flow minimum cut associated with the topology.

VI. Conclusion

We investigated the survivability of data over SONET/SDH, which is gaining increasing attention because, by such a mechanism, network operators can provide integrated data and voice services over their optical transport network to generate new revenue. We proposed two approaches for provisioning survivable DoS connections: PIVM, which is suitable for centralized implementation, and PREV, which is suitable for distributed implementation. Our approaches exploit the tradeoff between resource overbuild and fault-recovery time while utilizing the inverse-multiplexing capability of virtual concatenation to increase backup sharing. Our results demonstrated that PIVM achieves low resource overbuild and much faster fault recovery than that of data networks, and PREV achieves fast fault recovery comparable to SONET 50-ms protection (for typical U.S. backbone networks) while still achieving modest backup sharing.

We further investigated the impact of VCG size on network performance. We proved that it is NP-complete to compute a minimum-cost VCG having two members of combined bandwidth no less than a given value. Results from our effective heuristic showed that, by inversely multiplexing a connection into a few paths (depending on topology) for both working and backup VCGs, the network blocking performance is almost as good as the case in which a connection can be inversely multiplexed on an unlimited number of paths.

Appendix I
Proof of Theorem 1

Proof: As we have shown earlier, for a fixed U_Δ, the ILP reduces to a standard min-cost flow problem, and an optimal flow can be computed using a standard min-cost flow algorithm. Let f_0 be an optimal flow corresponding to a fixed U_Δ ($0 \leq U_\Delta \leq min\{B(p^b_{sd}), u\} - 2$). Let f_1 (and f_2) be an optimal flow corresponding to $U_\Delta + 1$ (and $U_\Delta + 2$).

By Theorem 3.7 (Augmenting Cycle Theorem) in [5], we can convert flow f_0 to flow f_1 by augmenting along a collection of cycles $\mathcal{C}_1$ in f_0's residual network with the capacity of some links increased by one. Let E_1 be the set of links whose capacity needs to be incremented. E_1 can be defined as

$$E_1 = \{\forall e \in E | B_a(e) > (U_\Delta + B_z(e))\}.$$

That is, E_1 includes those links which can carry more working flow when U_Δ increases. Using the same reasoning, we can convert flow f_0 to flow f_2 by augmenting along a collection of cycles $\mathcal{C}_2$ in f_0's residual networking with the capacity of the links in E_1 increased by one and the capacity of the links in E_2 increased by one more, where E_2 is defined as

$$E_2 = \{\forall e \in E | B_a(e) > (U_\Delta + B_z(e) + 1)\}.$$

Clearly, any of the cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$ have negative cost; otherwise, there is no point in augmenting the cycle. Since there is no cycle of negative cost in f_0's residual network without increasing the capacity of the links in E_1, any of the cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$ must contain at least one link from E_1 (note that $E_2 \subseteq E_1$). A possible solution for $\mathcal{C}_1$ is to choose all the cycles in $\mathcal{C}_2$ and augment by half of the capacity. Therefore, the drop in cost when we convert from flow f_0 to flow f_1 is at least half as much as that from flow f_0 to flow f_2, i.e.,

$$c_w(U_\Delta) - c_w(U_\Delta + 1) \geq \frac{1}{2} \cdot (c_w(U_\Delta) - c_w(U_\Delta + 2))$$

or

$$c_w(U_\Delta) - c_w(U_\Delta + 1) \geq c_w(U_\Delta + 1) - c_w(U_\Delta + 2)$$

Therefore, for $0 \leq U^1_\Delta < U^2_\Delta < min\{B(p^b_{sd}), u\}$,

$$\begin{aligned} c_w\left(U^1_\Delta\right) - c_w\left(U^1_\Delta + 1\right) &\geq c_w\left(U^1_\Delta + 1\right) - c_w\left(U^1_\Delta + 2\right) \\ &\geq \cdots \geq c_w\left(U^2_\Delta\right) - c_w\left(U^2_\Delta + 1\right) \end{aligned}$$

■

Appendix II
NP-Completeness of the Capacitated Minimum-Cost-Multi-Path (CMCMP) Problem

Basic idea: We reduce the NP-complete problem 3SAT [12] to the CMCMP problem.

For an arbitrary instance of 3SAT we construct a graph with a path p_1 corresponding to the false assignment of all the variables and a path p_2 corresponding to the true assignment of all the variables. The link costs and capacities are constructed so paths p_1 and p_2 are of combined capacity at least u and combined cost at most w if and only if the given instance of 3SAT is satisfiable.

Proof of Theorem 2: CMCMP $\in$ NP since a nondeterministic algorithm can guess two paths, p_1 and p_2, and check in polynomial time if these two paths are of combined capacity no less than u and combined cost no more than w.

Given a 3SAT instance F with clauses $\{D_1, D_2, \ldots, D_m\}$ and variables $Q = \{v_1, v_2, \ldots, v_n\}$, we construct in polynomial time an instance of CMCMP $G = (V, E, C, \lambda)$, B_a, s, and d,

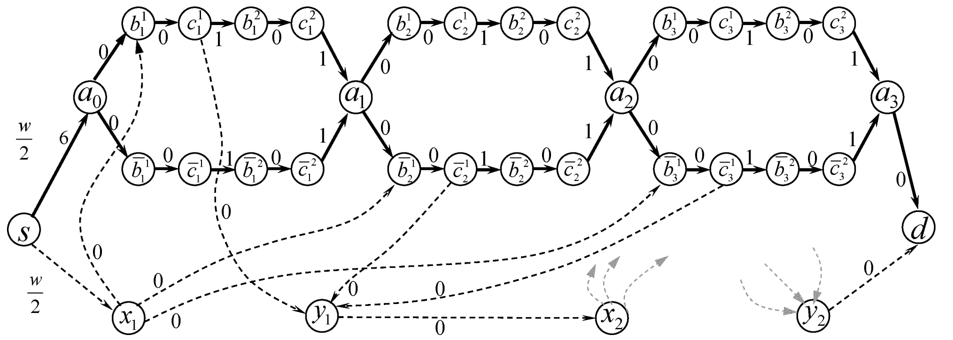


Fig. 11. Illustrative construction for a 3SAT instance $F = \{D_1, D_2\}, D_1 = v_1 \vee \bar{v}_2 \vee \bar{v}_3, D_2 = \bar{v}_1 \vee v_2 \vee v_3$, and $Q = \{v_1, v_2, v_3\}$ (the part corresponding to clause D_2 is not shown to avoid clutter). The number on every link is the cost. The dashed lines have one unit of available capacity, and the solid lines have $u - 1$ units of available capacity.

where $\lambda(e) = u$ for all $e \in E$. We now define the nodes V, the links E, the cost function C, and the available bandwidth function B_a. V has a source node s and a destination node d. The other nodes are in two groups. The first group is related to the variables in Q and has nodes a_k, b_j^i, $\bar{b}_j^i$, c_j^i, and $\bar{c}_j^i$, where $0 \le k \le n$, $1 \le i \le m$, $1 \le j \le n$. The second group of nodes is related to the clauses in F and has nodes x_i and y_i for $1 \le i \le m$.

The set of links (unidirectional fibers) E also has two groups. The first group of links will be used for assigning values to the variables in Q; the second group of links will be used for evaluating the Boolean value of the clauses. The links are:

— Group 1:

a) A link from node b_j^i to node c_j^i, $1 \le i \le m$ and $1 \le j \le n$.

b) A link from node $\bar{b}_j^i$ to node $\bar{c}_j^i$, $1 \le i \le m$ and $1 \le j \le n$.

c) A link from node c_j^i to node b_j^{i+1}, $1 \le i \le m-1$ and $1 \le j \le n$.

d) A link from node $\bar{c}_j^i$ to node $\bar{b}_j^{i+1}$, $1 \le i \le m-1$ and $1 \le j \le n$.

e) A link from node a_j to node b_{j+1}^1, and a link from node a_j to node $\bar{b}_{j+1}^1$, $0 \le j \le n-1$.

f) A link from node c_j^m to node a_j, and a link from node $\bar{c}_j^m$ to node a_j, $1 \le j \le n$.

— Group 2:

a) A link from node x_i to b_j^i, and a link from node c_j^i to node y_i, if and only if variable v_j is in clause D_i.

b) A link from node x_i to $\bar{b}_j^i$, and a link from node $\bar{c}_j^i$ to node y_i, if and only if $\bar{v}_j$ is in clause D_i.

c) A link from node y_i to node x_{i+1}, $1 \le i \le m-1$.

In addition to the above links, we also have a link from node s to node a_0, a link from node a_n to node d, a link from node s to node x_1, and a link from node y_m to node d.

The link-cost function, $C : E \to Z$, is defined as follows:

$$C(e) = \begin{cases} \left\lceil \frac{w}{2} \right\rceil - m \cdot n, & \text{if } e = \langle s, a_0 \rangle \\ \left\lfloor \frac{w}{2} \right\rfloor, & \text{if } e = \langle s, x_1 \rangle \\ 1, & \text{if } e = \left\langle c_j^i, b_j^{i+1} \right\rangle, \text{ or } e = \left\langle \bar{c}_j^i, \bar{b}_j^{i+1} \right\rangle, \\ & \text{or } e = \left\langle c_j^m, a_j \right\rangle, \text{ or } e = \left\langle \bar{c}_j^m, a_j \right\rangle \\ & (1 \le i \le m-1 \text{ and } 1 \le j \le n) \\ 0, & \text{otherwise.} \end{cases}$$

The available bandwidth function, $B_a : E \to Z$, is defined as follows:

$$B_a(e) = \begin{cases} 1, & \text{if } e \text{ originates or terminates at node } x_i \text{ or} \\ & \text{node } y_i (1 \le i \le m) \\ u-1, & \text{otherwise} \end{cases}$$

It is easy to see that the construction can be done in polynomial time. An illustrative construction (the construction for the second clause is similar to the first one and is not shown) for a 3SAT instance $F = \{D_1, D_2\}$, $D_1 = v_1 \vee \bar{v}_2 \vee \bar{v}_3$, $D_2 = \bar{v}_1 \vee v_2 \vee v_3$, and $Q = \{v_1, v_2, v_3\}$ is shown in Fig. 11.

We now show that, if F is satisfiable, then from node s to node d in graph G there exist two paths of combined cost w and combined bandwidth u. Let $v_1 = z_1, v_2 = z_2, \ldots, v_n = z_n$ be an assignment that satisfies F, where $z_j \in \{0, 1\}$ for $1 \le j \le n$. The two paths can be routed as follows. The first path, p_1, is routed via the nodes b_j^i and c_j^i ($1 \le i \le m$ and $1 \le j \le n$) if and only if $z_j = 0$; otherwise, p_1 is routed via nodes $\bar{b}_j^i$ and $\bar{c}_j^i$ (p_1 will also need to traverse all the a_k nodes for $0 \le k \le n$). The second path, p_2, is routed via x_i, y_i, and other nodes defined as follows. By the construction, the path from node x_i to node y_i corresponds to clause D_i, $|D_i| = 3$. Without loss of generality, let $D_i = v_f \vee \bar{v}_g \vee v_h$, where f, g, and h are distinct integers between 1 and n. Then, there will be three paths from node x_i to node y_i that go through nodes b_f^i, $\bar{b}_g^i$, and b_h^i, respectively. Since $v_1 = z_1, v_2 = z_2, \ldots, v_n = z_n$ is an assignment that satisfies F, either $z_f = 1$, or $z_g = 0$, or $z_h = 1$. If $z_f = 1$, then path p_2 traverses links $\langle b_f^i, c_f^i \rangle$ and $\langle c_f^i, y_i \rangle$; if $z_g = 0$, then path p_2 traverses links $\langle \bar{b}_g^i, \bar{c}_g^i \rangle$ and $\langle \bar{c}_g^i, y_i \rangle$; if $z_h = 1$, then path p_2 traverses links $\langle b_h^i, c_h^i \rangle$ and $\langle c_h^i, y_i \rangle$; if more than one condition is true, randomly pick one.

Paths p_1 and p_2 so selected have combined bandwidth u and combined cost w due to the following facts:

1) The capacity of path p_2 is unity.

2) The capacity of path p_1 is $u - 1$ because p_1 is link-disjoint to p_2, which can be shown as follows. Path p_1 only traverses links in Group 1, and the Group-1 links p_1 traverses correspond to the literals of value 0. Path p_2 traverses links in both Group 1 and Group 2, and the Group-1 links p_2 traverses correspond to the literals of value 1. By the construction, a Group-1 link corresponding to a literal of value 0 is disjoint to a Group-1 link corresponding to a literal of value 1; and a Group-1 link is disjoint to a

Group-2 link. Therefore, paths p_1 and p_2 are link-disjoint. Because all the links path p_1 traverses have capacity $u-1$, the capacity of p_1 is $u-1$.

3) The cost of path p_2 is $\lfloor w/2 \rfloor$ because all the Group-1 links p_2 traverses have the cost of zero.

4) The cost of path p_1 is $\lceil w/2 \rceil$ because: (i) p_1 traverses link $\langle s, a_0 \rangle$, which has cost $\lceil w/2 \rceil - m \cdot n$. (ii) p_1 traverses either links $\langle c_j^i, b_j^{i+1} \rangle$ and $\langle c_j^m, a_j \rangle$ or links $\langle \bar{c}_j^i, \bar{b}_j^{i+1} \rangle$ and $\langle \bar{c}_j^m, a_j \rangle$ ($1 \le i \le m-1$ and $1 \le j \le n$), all of which have unity cost. The total cost of the links in this category is $(m-1) \cdot n + n = m \cdot n$. (iii) All the other links p_1 traverses are of zero cost.

Thus, we have shown that, if F is satisfiable, then we can find two paths of combined bandwidth u and combined cost w.

We now show that, if there are two paths p_1 and p_2 from node s to node d of combined bandwidth u and combined cost w in graph G, then F is satisfiable due to the following facts.

1) Since the amount of outgoing flow at node s is u, both links $\langle s, a_0 \rangle$ and $\langle s, x_1 \rangle$ will be used by paths p_1 and p_2. Without loss of generality, suppose that path p_1 traverses link $\langle s, a_0 \rangle$, and path p_2 traverses link $\langle s, x_1 \rangle$.

2) The capacity of path p_1 is $u-1$ and the capacity of path p_2 is unity because: (i) the combined capacity of paths p_1 and p_2 is u; (ii) path p_1 traverses link $\langle s, a_0 \rangle$, which has available capacity $u-1$; (iii) path p_2 traverses link $\langle s, x_1 \rangle$, which has unity available capacity.

3) Path p_1 can only traverse links in Group 1; otherwise, the capacity of path p_1 is smaller than $u-1$ and the combined capacity of the two paths is smaller than u.

4) Paths p_1 and p_2 are link-disjoint because the capacity of any of the Group-1 links is $u-1$.

5) Path p_1 traverses either links $\langle c_j^i, b_j^{i+1} \rangle$ and $\langle c_j^m, a_j \rangle$ or links $\langle \bar{c}_j^i, \bar{b}_j^{i+1} \rangle$ and $\langle \bar{c}_j^m, a_j \rangle$ ($1 \le i \le m-1$ and $1 \le j \le n$), all of which have unity cost. The total cost of the links in this category is $(m-1) \cdot n + n = m \cdot n$. Therefore, the cost of path p_1 is $\lceil w/2 \rceil$.

6) The cost of path p_2 can be no more than $\lfloor w/2 \rfloor$ because the combined cost of paths p_1 and p_2 is w. Path p_2 can not traverse any Group-1 links of nonzero cost because p_2 traverses link $\langle s, x_1 \rangle$ of cost $\lfloor w/2 \rfloor$.

7) We represent the route for path p_1 by a n-bit binary number, $z_1 z_2 \ldots z_n$. Path p_1 is routed as follows: if bit z_j is 1, then p_1 is routed over the nodes $\bar{b}_j^i$; otherwise, it is routed over the nodes b_j^i. Note that there is a one-to-one mapping between an n-bit binary number and a path using only Group-1 links in the network. Let $z_1 z_2 \ldots z_n$ be the n-bit number that corresponds to path p_1. Then, there is a path from node x_i to node y_i that does not use the links traversed by p_1, if and only if clause D_i is true under the following assignment: $v_j = \bar{z}_j$, where $1 \le j \le n$. The reason is as follows. Without loss of generality, let $D_i = v_f \vee \bar{v}_g \vee v_h$, where f, g, and h are distinct integers between 1 and n. There are three paths from node x_i to node y_i that traverse nodes b_f^i, $\bar{b}_g^i$, and b_h^i (note that path p_2 cannot traverse any Group-1 links of nonzero cost). Path p_1 will traverse the nodes b_f^i, $\bar{b}_g^i$, and b_h^i, if and only if $z_f = 0$, $z_g = 1$, and $z_h = 0$, respectively. Thus, there exists a path from node x_i to y_i if and only if $z_f = 1$, or $z_g = 0$, or $z_h = 1$, which is exactly the condition under which D_i will be true.

Since paths p_1 and p_2 have combined capacity u and combined cost w, D_i is true for all $1 \le i \le m$ by the above facts. Consequently, F is satisfiable for the assignment: $v_j = z_j$, $1 \le j \le n$, where $z_1 z_2, \ldots, z_n$ is the n-bit number corresponding to path p_1. Thus, CMCMP is NP-complete.

References

[1] "Synchronous Optical Networks (SONET)," ANSI T1X1.5, 2001-062, Jan. 2001.

[2] "Metro Ethernet Networks—A Technical Overview," Metro Ethernet Forum white paper, Jul. 2002.

[3] "Link Capacity Adjustment Scheme (LCAS) for Virtual Concatenated Signals," ITU-T Rec. G.7042/Y.1305, Nov. 2001.

[4] "Network Node Interface for the Synchronous Digital Hierarchy (SDH)," ITU-T Rec. G.707, Apr. 2002.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[6] E. Bouillet, J.-F. Labourdette, R. Ramamurthy, and S. Chaudhuri, "Enhanced algorithm cost model to control tradeoffs in provisioning shared mesh restored lightpaths," presented at the Optical Fiber Communication Conf., Anaheim, CA, Mar. 2002, paper ThW2.

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

[8] B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang, "Optical network design and restoration," *Bell Labs Tech. J.*, vol. 4, pp. 58–84, Jan.–Mar. 1999.

[9] S. Fortune, J. E. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *J. Theoretical Comput. Sci.*, vol. 10, no. 2, pp. 111–121, 1980.

[10] A. Fumagalli, I. Cerutti, and M. Tacca, "Optimal design of survivable mesh networks based on line switched WDM self-healing rings," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 501–512, Jun. 2003.

[11] A. Fumagalli and L. Valcarenghi, "IP restoration versus WDM protection: is there an optimal choice?," *IEEE Network*, vol. 14, pp. 34–41, Nov.–Dec. 2000.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman Co., 1979.

[13] O. Gerstel and R. Ramaswami, "Optical layer survivability: a services perspective," *IEEE Commun. Mag.*, vol. 38, no. 3, pp. 104–113, Mar. 2000.

[14] M. Goyal, G. Li, and J. Yates, "Shared mesh restoration: a simulation study,," in *Proc. OFC*, Mar. 2002, pp. 489–490.

[15] W. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration," in *Proc. IEEE ICC*, vol. 1, 1998, pp. 537–543.

[16] O. Hauser, M. Kodialam, and T. V. Lakshman, "Capacity design of fast path restorable optical networks," in *Proc. IEEE INFOCOM*, vol. 2, Jun. 2002, pp. 817–826.

[17] E. Hernandez-Valencia, M. Scholten, and Z. Zhu, "The generic framing procedure (GFP): an overview," *IEEE Commun. Mag.*, vol. 40, pp. 63–71, May 2002.

[18] S. Koo and S. Subramaniam, "Trade-offs between speed, capacity, and restorability in optical mesh network restoration," in *Proc. OFC*, Mar. 2002, pp. 487–489.

[19] M. Kodialam and T. V. Lakshman, "Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 399–410, Jun. 2003.

[20] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, "Efficient distributed path selection for shared restoration connections," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 140–149.

[21] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," in *Proc. IEEE INFOCOM*, vol. 2, Apr. 2001, pp. 699–708.

[22] R. MacDonald, L.-P. Chen, C.-X. Shi, and B. Faer, "Requirements of optical layer network restoration," in *Proc. OFC*, Mar. 2000, pp. 68–70.

[23] M. Medard, R. A. Barry, S. Finn, W. He, and S. Lumetta, "Generalized loop-back recovery in optical mesh networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 153–164, Feb. 2002.

[24] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: a new approach to the design of WDM-based networks," *IEEE J. Select. Areas Commun.*, vol. 20, no. 5, pp. 800–809, May 2002.

[25] G. Mohan, C. S. R. Murthy, and A. K. Somani, "Efficient algorithms for routing dependable connections in WDM optical networks," *IEEE/ACM Trans. Netw.*, vol. 9, no. 10, pp. 553–566, Oct. 2001.

[26] B. Mukherjee, *Optical Communication Networks*. New York: McGraw-Hill, 1997.

[27] C. Ou, J. Zhang, H. Zang, L. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in WDM mesh networks," *J. Lightw. Technol.*, vol. 22, no. 5, pp. 1223–1232, May 2004.

[28] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM mesh networks," *J. Lightw. Technol.*, vol. 21, no. 4, pp. 870–883, Apr. 2003.

[29] N. Robertson and P. D. Seymour, "Graph minors XIII: the disjoint paths problem," *J. Combin. Theory Ser. B*, vol. 63, pp. 65–110, 1995.

[30] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee, "Fault tolerance in IP-over-WDM networking: WDM protection versus IP restoration," *IEEE J. Select. Areas Commun.*, vol. 20, no. 1, pp. 21–33, Jan. 2002.

[31] P. Sebos, J. Yates, G. Hjlmtsson, and A. Greenberg, "Auto-discovery of shared risk link groups," in *Proc. OFC*, vol. 3, 2001, pp. WDD3-1–WDD3-3.

[32] S. Stanley, "Making SONET Ethernet-Friendly," Lightreading report, Mar. 2003.

[33] J. Strand, A. Chiu, and R. Tkach, "Issues for routing in the optical layer," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 81–87, Feb. 2001.

[34] C. Su and X. Su, "Protection path routing on WDM networks," in *Proc. OFC*, vol. 2, Mar. 2001, pp. TuO2-T1–TuO2-T3.

[35] X. Su and C. Su, "An online distributed protection algorithm in WDM networks," in *Proc. IEEE ICC*, vol. 5, Jun. 2001, pp. 1571–1575.

[36] 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.

[37] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Trap avoidance and protection schemes in networks with shared risk link groups," *J. Lightw. Technol.*, vol. 21, no. 11, pp. 2683–2693, Nov. 2003.

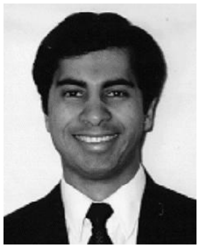
[38] H. Zang, C. Ou, and B. Mukherjee, "Path-protection routing and wavelength-assignment (RWA) in WDM mesh networks under duct-layer constraints," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 248–258, Apr. 2003.

[39] K. Zhu, H. Zang, and B. Mukherjee, "Exploiting the benefit of virtual concatenation technique to the optical transport networks," in *Proc. OFC*, Mar. 2003, pp. 363–364.

Canhui (Sam) Ou (S'02) received the B.S. degree from Peking University, Beijing, China, in 2000, and the M.S. and Ph.D. degrees from the University of California, Davis, in 2001 and 2004, respectively.

He is currently a Principal Member of Technical Staff at AT&T Services, Inc., San Ramon, CA. His technical interests include WDM networks, MPLS, optical Ethernet, IPTV, and FTTx.

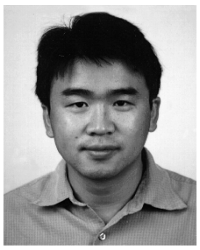
Dr. Ou is a coauthor of the book *Survivable Optical WDM Networks* (Springer, 2005).



Laxman H. Sahasrabuddhe received the B.Tech. degree from the Indian Institute of Technology, Kanpur, India, in 1992, the M.Tech. degree from the Indian Institute of Technology, Madras, India, in 1994, and the Ph.D. degree from the University of California, Davis, in 1999.

Currently, he is a Technology Specialist with Park, Vaughan & Fleming LLP, Davis, CA.

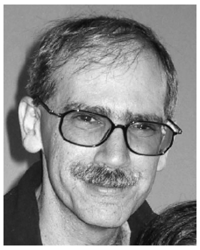
Dr. Sahasrabuddhe is the recipient of the Best Doctoral Dissertation Award, from the College of Engineering, University of California, Davis, for his research on WDM Optical Networks.



Keyao Zhu (S'98) received the B.S. degree from Peking University, Beijing, China, in 1998, and the M.S. and Ph.D. degree from the University of California, Davis, in July 2000 and September 2003, respectively.

From August 2003 to September 2004, he was with Research and Innovation, Alcatel Shanghai Bell. Currently, he is with Brion Tech. Inc., Santa Clara, CA.

Dr. Zhu serves or has served as a Technical Committee Member of ICC'04 and ICC'05. He is the recipient of the Best Doctoral Dissertation Award from the College of Engineering, University of California, Davis, for his research on WDM optical networks.



Charles U. Martel received the B.S. degree in computer science from the Massachusetts Institute of Technology, Cambridge, in 1975 and the Ph.D. degree in computer science from the University of California at Berkeley in 1980.

Since 1980, he has been on the Computer Science faculty at the University of California at Davis, where he is a founder and former chair of the Computer Science Department. His research interests include algorithms for parallel and distributed systems, scheduling, graph theory, and network and security algorithms.

Dr. Martel has been a member of the Association for Computing Machinery (ACM) since 1980.

Biswanath Mukherjee (S'82–M'87) received the B.Tech. (Hons) degree from the Indian Institute of Technology, Kharagpur, India, in 1980 and the Ph.D. degree from the University of Washington, Seattle, in 1987. While at the University of Washington, he held a GTE Teaching Fellowship and a General Electric Foundation Fellowship.

In 1987, he joined the University of California, Davis, where he has been a Professor of computer science since 1995. He served as Chairman of the Department of Computer Science from 1997 to 2000. He is a Member of the Board of Directors of IPLocks Inc., a Silicon Valley startup company. He has consulted for and served on the Technical Advisory Board of a number of startup companies in optical networking. He has served on the editorial boards of the *ACM/Baltzer Wireless Information Networks (WINET)*, the *Journal of High-Speed Networks*, *Photonic Network Communications*, and *Optical Network Magazine*. He is the author of the textbook *Optical Communication Networks* (McGraw-Hill, 1997), a book which received the Association of American Publishers, Inc. 1997 Honorable Mention in Computer Science. His research interests include lightwave networks, network security, and wireless networks.

Dr. Mukherjee is a corecipient of paper awards presented at the 1991 and 1994 National Computer Security Conferences. He is a winner of the 2004 Distinguished Graduate Mentoring Award from UC Davis. He has served on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING and *IEEE Network*, and has also served as Editor-at-Large for optical networking and communications for the IEEE Communications Society. He served as the Technical Program Chair of the IEEE INFOCOM 1996 conference, and also serves as the Chairman of the IEEE Communication Society's Optical Networking Technical Committee (ONTC).