Generalized Loop-Back Recovery in Optical Mesh Networks

Muriel Médard, *Member, IEEE*, Richard A. Barry, *Member, IEEE*, Steven G. Finn, *Member, IEEE*, Wenbo He, and Steven S. Lumetta, *Member, IEEE*

***Abstract*—Current means of providing loop-back recovery, which is widely used in SONET, rely on ring topologies, or on overlaying logical ring topologies upon physical meshes. Loop-back is desirable to provide rapid preplanned recovery of link or node failures in a bandwidth-efficient distributed manner. We introduce generalized loop-back, a novel scheme for performing loop-back in optical mesh networks. We present an algorithm to perform recovery for link failure and one to perform generalized loop-back recovery for node failure. We illustrate the operation of both algorithms, prove their validity, and present a network management protocol algorithm, which enables distributed operation for link or node failure. We present three different applications of generalized loop-back. First, we present heuristic algorithms for selecting recovery graphs, which maintain short maximum and average lengths of recovery paths. Second, we present WDM-based loop-back recovery for optical networks where wavelengths are used to back up other wavelengths. We compare, for WDM-based loop-back, the operation of generalized loop-back operation with known ring-based ways of providing loop-back recovery over mesh networks. Finally, we introduce the use of generalized loop-back to provide recovery in a way that allows dynamic choice of routes over preplanned directions.**

***Index Terms*—WDM, loop-back, network restoration, mesh networks.**

I. Introduction

FOR WDM networks to offer reliable high-bandwidth services, automatic self-healing capabilities, similar to those provided by SONET, are required. In particular, preplanned ultrafast restoration of service after failure of a link or node is required. As WDM networks mature and expand, the need has emerged for self-healing schemes which operate over a variety of network topologies and in a manner which is bandwidth efficient. While SONET provides a known and robust means of providing recovery in high-speed networks, the techniques used for SONET are not always immediately applicable to WDM systems. Certain issues, such as wavelength assignment and wavelength changing, make WDM self-healing different from SONET self-healing. Our purpose is to present a method for service restoration in optical networks with the following characteristics.

- *Speed*: We want the speed of recovery to be of the order of the speed of switching and, therefore, require our algorithms to have minimal real-time processing overhead.
- *Transparency*: We seek a method of recovery which can be done at the optical layer, without regard for whatever protocol(s) may be running over the optical layer.
- *Flexibility*: Our method should not constrain primary routings and should provide a large choice of back-up routes to satisfy such requirements as bounds on average or maximum back-up length.

In this paper, we present an approach that altogether moves away from rings. The rationale behind our approach is that, while ring recovery makes sense over network topologies that are composed of interconnected rings, rings are not fundamental to network recovery over mesh networks. Indeed, even embedding rings over a given topology can have significant implications for hardware costs [1]. We present generalized loop-back, a new method of achieving loop-back recovery over arbitrary two-link-redundant and two-node-redundant networks to restore service after the failure or a link or a node, respectively. A two-link (node) redundant network remains connected after failure of a link (node). We abbreviate two-link (node) redundant by link (node) redundant. Loop-back recovery over mesh networks without the use of ring covers was first introduced in [7] and [20]. We represent each network by a graph, with each node corresponding to a vertex and each link (which may contain several fibers) to an undirected edge. Failure of a link or node is mapped to the disappearance of an edge or vertex in the corresponding graph. The principle behind generalized loop-back is to create primary and secondary digraphs so that, upon failure of a link or node in the network, the secondary digraph can be used to carry back-up traffic that provides loop-back to the primary graph. Each primary or secondary digraph may correspond to a wavelength, a set of wavelengths, a fiber, or a set of full fibers. The secondary digraph is the conjugate of the primary digraph. Each direction in a link is associated with a given primary graph. Our algorithms perform the choice of directions to establish our digraphs.

Our approach meets our three goals, i.e., speed, transparency, and flexibility. Although we use preplanning of directions, our

Manuscript received June 7, 2000; revised April 12, 2001; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Rouskas. This work was supported in part by the Defense Advanced Research Projects Agency under Grant MDA972-99-1-0005.

M. Médard is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: medard@mit.edu).

R. A. Barry is with Sycamore Networks Inc., Chelmsford, MA 01824 USA (e-mail: rick@sycamorenet.com).

S. G. Finn is with the Advanced Networks Group, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02173 USA (e-mail: finn@ll.mit.edu).

W. He is with Cisco Systems Inc., Champaign, IL 61820 USA and is also with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: wenbohe@students.uiuc.edu).

S. S. Lumetta is with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: steve@crhc.uiuc.edu).

Publisher Item Identifier S 1063-6692(02)01078-6.

network management protocol determines, in real time, the back-up route that will be utilized. We do not, however, require processing as in traditional dynamic recovery schemes. In effect, **our network management protocol provides dynamic real-time discovery of routings along preplanned directions determined by our direction selection algorithms**. Since our protocol (see Section III-C) requires very simple processing and the optical layer remains responsible for recovery (ensuring transparency), we have speed of recovery combined with flexibility. In particular, depending on availability of links or wavelengths (which may be affected by congestion or failures in the network), different back-up routes may be selected, but the selection will be automatic and will not require active comparison, by the network management, of the different possible routes.

In Section I-A, we give an overview of relevant work in the area of network protection and restoration. In Section II-A, we discuss generalized loop-back recovery for link failure in arbitrary link-redundant networks. In Section II-B, we present our method for loop-back recovery for node failure in arbitrary node-redundant networks. A simple network protocol, presented in Section II-C, allows for distributed operation of recovery from link or node failure.

Section III considers a range of different applications for generalized loop-back as follows.

- We address the goal of flexibility in the choice of back-up routings. We present a means of selecting directions for generalized loop-back so as to avoid excessive path lengths. Our algorithm allows a choice among a large number of alternative directions. The choice of directions may greatly affect the length of back-up paths. We present heuristic algorithms that significantly reduce the average and maximum length of recovery paths over random choices of directions.
- We may use generalized loop-back for wavelength-based recovery, which we term WDM-based loop-back recovery, instead of fiber-based recovery in mesh networks. We illustrate why the method of cover of rings using double-cycle covers is not directly applicable to WDM loop-back recovery.
- Generalized loop-back can yield several choices of backup routes for a given set of directions. We briefly illustrate how generalized loop-back can be used to avoid the use of certain links.

Finally, in Section IV, we present conclusions and directions for further research.

A. Background

Methods commonly employed for link protection in high-speed networks can be classified as either dynamic or preplanned, though some hybrids schemes also exist. Dynamic restoration typically involves a search for a free path using back-up capacity through broadcasting of help messages. Overheads due to message passing and software processing render dynamic processing slow. For dynamic link restoration using digital cross-connect systems, a two second restoration time is a common goal for SONET. Preplanned methods depend mostly on lookup tables and switches or add–drop multiplexers. To meet our speed requirement, we consider preplanned methods, even though they may suffer from poorer capacity utilization than dynamic systems, which use real-time availability of back-up capacity.

Within preplanned methods, we distinguish between path and link or node restoration. Path restoration refers to recovery applied to connections following a particular path across a network. Link or node restoration refers to recovery of all the traffic across a failed link or node, respectively. Path restoration may be itself subdivided into two different types: live (dual-fed) back-up, and event-triggered back-up. In the first case, two live flows, a primary and a back-up, are transmitted. The two flows are link-disjoint if we seek to protect against link failure, or node-disjoint (except for the end nodes) if we seek to protect against node failure. Upon failure of a link or node on the primary flow, the receiver switches to receiving on the back-up. Recovery is thus extremely fast, requiring action only from the receiving node, but back-up capacity is not shared among connections. In the second case, event-triggered path restoration, the back-up path is only activated when a failure occurs on a link or node along the primary path. Backup capacity can be shared among different paths [36], thus improving capacity utilization for back-up channels and allowing for judicious planning. However, recovery involves coordination between sender and receiver after a failure and action from nodes along the back-up path. This coordination may lead to delays and management overhead.

Preplanned link or node restoration can be viewed as a compromise between live and event-triggered path restoration. Preplanned link restoration is not as capacity-efficient as event-triggered path restoration, but is more efficient than live back-up path restoration since sharing of back-up bandwidth is allowed. The traffic along a failed link or node is recovered, without consideration for the end points of the traffic carried by the link or node. Thus, only the two nodes adjacent to the failure need to engage in recovery. The back-up is not live, but triggered by a failure. Overviews of the different types of protection and restoration methods and comparison of the tradeoffs among them can be found in [2], [16], [18], [25], [26], and [40].

Link or node restoration also benefits from a further advantage, which makes it very attractive for preplanned recovery: **since it is not dependent upon specific traffic patterns, it can be preplanned once and for all**. Thus, link or node restoration is particularly attractive at lower layers, where network management may not be aware, at all locations of the network, of the origination and destination, or of the format [39] of all the traffic being carried at that location. Therefore, in this paper, we concentrate on preplanned link and node restoration in order to satisfy our transparency requirement. Moreover, link restoration satisfies the first part of our flexibility goal since restoration is done without consideration for primary routings.

For preplanned link restoration, the main approaches have involved covers of rings and, more recently, preplanned cycles [11]. The most direct approach is to design the network in term of rings. The building blocks of SONET networks are generally self-healing rings (SHRs) and diversity protection

(DP) [32], [31], [38]. SHRs are unidirectional path-switched rings (UPSRs) or bidirectional line-switched rings (BLSRs), while DP refers to physical redundancy where a spare link (node) is assigned to one or several links (nodes) ([38, pp. 315–332]). In rings, such as BLSR, link or node restoration is simply implemented using loop-back, which we explain in Section II.

Ring-based architectures may be more expensive than meshes [1], [35] and, as nodes are added, or networks are interconnected, ring-based structures may be difficult to preserve, thus limiting their scalability [34], [35], [38]. However, rings are not necessary to construct survivable networks [24], [33]. Mesh-based topologies can also provide redundancy [17], [28], [34]. For reasons of cost and extensibility, mesh-based architectures are more promising than interconnected rings.

Covering mesh topologies with rings is a means of providing both mesh topologies and distributed ring-based restoration. There are several approaches to covers of rings for networks in order to ensure link restorability. One approach is to cover nodes in the network by rings [31]. In this manner, a portion of links are covered by rings. If primary routings are restricted to the covered links, then link restoration can be effected on each ring in the same manner as in a traditional SHR, by routing the back-up traffic around the ring in the opposite direction to the primary traffic. Using such an approach, the uncovered links can be used to carry unprotected traffic, i.e., traffic which may not be restored if the link that carries it fails.

To allow every link to carry protected traffic, other ring-based approaches ensure every link is covered by a ring. One approach to selecting such covers is to cover a network with rings so that every link is part of at least one ring [10]. This approach suffers from some capacity drawbacks. With fiber-based restoration, every ring is a four-fiber ring. A link covered by two rings requires eight fibers; a link covered by n rings requires $4n$ fibers. Alternatively, the logical fibers can be physically routed through four physical fibers, but only at the cost of significant network management overhead. Minimizing the amount of fiber required to obtain redundancy using ring covers is equivalent to finding the minimum cycle cover of a graph, an NP-complete problem [13], [30], although bounds on the total length of the cycle cover may be found [5].

A second approach to ring covers, intended to overcome the difficulties of the first approach, is to cover every link with exactly two rings, each with two fibers. The ability to perform loop-back style restoration over mesh topologies was first introduced in [3] and [4]. In particular, [4] considers link failure restoration in optical networks with arbitrary two-link redundant arbitrary mesh topologies and bidirectional links. The approach is an application of the double-cycle ring cover [15], [27], [29]. For planar graphs, the problem can be solved in polynomial time; for nonplanar graphs, it is conjectured that double-cycle covers exist, and a counterexample would have to obey certain properties [9]. Node recovery can be effected with double-cycle ring covers, but such restoration requires cumbersome hopping among rings. In Section III-B, we consider double-cycle covers in the context of wavelength-based recovery.

In order to avoid the limitations of ring covers, an approach using preconfigured cycles, or p-cycles, is given in [11]. A p-cycle is a cycle on a redundant mesh network. Links on the p-cycle are recovered by using the p-cycle as a conventional BLSR. Links not on the p-cycle are recovered by selecting, along the p-cycle, a path connecting the nodes, which are the end-points of the failed link. Note that some difficulty arises from the fact that several p-cycles may be required to cover a network, making management among p-cycles necessary. A single p-cycle may be insufficient because a Hamiltonian might not exist, even in a two-connected graph. Even finding p-cycles which cover a large number of nodes, may be difficult. Some results [8], [14], [41] and conjectures [12], [37] exist concerning the length of maximal cycles in two-connected graphs. The p-cycle approach is in effect a hybrid ring approach, which mixes path restoration (for links not on the p-cycle) with ring recovery (for links on the p-cycle).

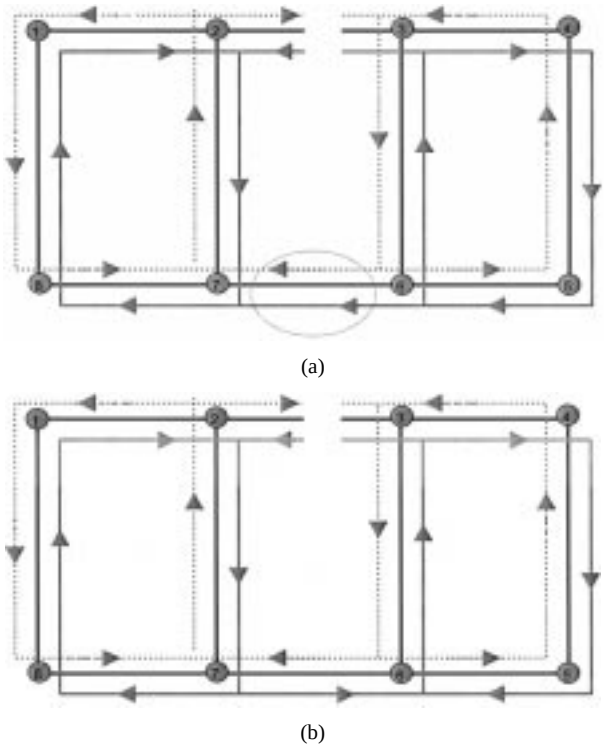
II. Generalized Loop-Back

A. Generalized Loop-Back for Recovery From Link Failures

The gist of our approach is to eliminate the use of rings. Instead, a primary (secondary) digraph (corresponding to a set of unidirectional fibers or wavelengths) is backed up by another secondary (primary) digraph (corresponding to a set of unidirectional fibers or wavelengths in the reverse direction of the primary (secondary) digraph). After a failure occurs, we broadcast the stream carried by the primary (secondary) digraph along the failed link onto the secondary (primary) digraph. We later show a protocol which ensures that only a single connection arrives to each node on the back-up path. When the back-up path reaches the node that lost its connection along the primary (secondary) digraph because of the failure, the traffic is restored onto the primary (secondary) digraph.

To illustrate our method, consider a simple sample network. Our algorithm works by assigning directions to each of the two fibers on each link. Fig. 1(b) shows in full arrow lines the directions of the primary digraph for each link and in thin dashed arrow lines the directions of the secondary digraph for each link. The topology of the network is shown in bold lines without arrows. A break in a link is shown by discontinued lines. The shortest back-up path is node 3 $\rightarrow$ node 6 $\rightarrow$ node 5 $\rightarrow$ node 4. Node 3 eliminates a duplicate connection that arrives to it via node 6 $\rightarrow$ node 5 $\rightarrow$ node 4 $\rightarrow$ node 3. Node 7 eliminates a duplicate connection that arrives to it via node 2 $\rightarrow$ node 1 $\rightarrow$ node 8 $\rightarrow$ node 7. Back-haul need not always occur. For instance, in Fig. 1(b), if the original connection went from node 4 to node 2 via node 3, then after recovery, the connection would commence at node 4 and traverse, in order, nodes 3, 6, 7 en route to node 2.

Not every assignment of directions provides the possibility for loop-back recovery. As an example, consider in Fig. 1(a) the same network topology as in Fig. 1(b) with different directions. The directions are provided in such a way that, when no failures are present, all nodes are reachable from each other on the primary wavelength on fiber 1 and on the secondary wavelength on fiber 2. However, the same link failure as in Fig. 1(b) is not recoverable. This example illustrates the importance of proper selection of the directions on the links.



(a)

(b)

Fig. 1. Example of choice of directions on links. In (a), the directions are chosen with our algorithm and offer as well as loop-back recovery in the case of a link failure. In (b), the directions provide connectivity when there are no failures, but do not enable recovery.

We may now formalize our approach. We define an undirected graph $G = (N, E)$ to be a set of nodes N and edges E. With each edge $[x, y]$ of an undirected graph, we associate two directed arcs (x, y) and (y, x). We assume that, if an edge $[x, y]$ fails, then arcs (x, y) and (y, x) both fail. A directed graph $P = (N, A)$ is a set of nodes N and a set of directed arcs A. Given a set of directed arcs A, define the reversal of A to be $\underline{A} = \{(i, j) \,|\, (j, i) \in A\}$. Similarly, given any directed graph $P = (N, A)$, define $\underline{P} = (N, \underline{A})$ to be the reversal of P.

Let us consider that we have a two-vertex (edge)-connected graph, or redundant graph $G = (N, E)$, i.e., removal of a vertex (edge) leaves the graph connected. Our method is based on the construction of a pair of directed spanning sub-graphs $B = (N, A)$ and $R = \underline{B} = (N, \underline{A})$, each of which can be used for primary connections between any pair of nodes in the graph. In the event of a failure, connections on y are looped back around the failure using R. Similarly, connections on R are looped back around the failure using B. For instance, if G were a ring, then B and R would be the clockwise and counterclockwise cycles around the ring.

To see how loop-back operates in a general mesh network, consider first the case where an edge $[x, y]$ fails. Assume (w, y) and (x, z) are arcs of R and that the shortest loop-back path around $[x, y]$ is node $x \rightarrow$ node $z \rightarrow \cdots \rightarrow$ node $w \rightarrow$ node y. We create two looping arcs $\mathrm{Bloop}_{x,z}$ and $\mathrm{Rloop}_{y,w}$. $\mathrm{Bloop}_{x,z}$ is created at node x by attaching the tail of $(z, x) \in A$ to the head of $(x, z) \in \underline{A}$ so that signals that arrive for transmission on (x, y) in B are now looped back at x to R. Similarly, $\mathrm{Rloop}_{y,w}$ is created by attaching the tail of $(w, y) \in \underline{A}$ to the head of $(y, w) \in A$ so that any signal that arrives for transmission on (w, y) in R is looped back to B at y. Fig. 2 illustrates our loop-back example. Edge $[x, y]$ can be successfully bypassed as long as there exists a working path with sufficient capacity from x to y in R and a working path with sufficient capacity from y to x in B.

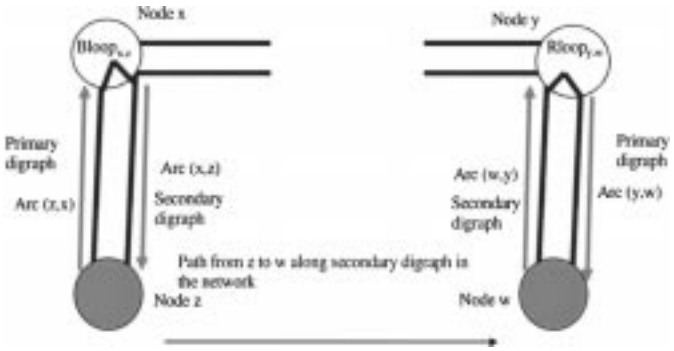


Fig. 2. Generalized loop-back mechanism.

Let us consider that we have an edge-redundant undirected graph $G = (N, E)$. We seek a directed spanning sub-graph $B = (N, A)$ of G, and its associated reversal $R = \underline{B} = (N, \underline{A})$, such that the following occurs.

- **Condition 1:** B is connected, i.e., there is a directed path in B between all nodes.
- **Condition 2:** $(x, y) \in A \Rightarrow (y, x) \notin A$.

Since R is connected if and only if B is connected, Condition 1 ensures that any connection can be routed on B or R. Condition 1 also ensures that loop-back can be performed. Suppose edge $[x, y]$ fails. Also suppose without loss of generality that (x, y) is an arc of B. In order to effect loop-back, we require that there exist a path from x to y in $R \backslash (y, x)$ and a path from y to x in $B \backslash (x, y)$. Such paths are guaranteed to exist because B and R are connected and because the path from x to y in B (y to x in R) obviously does not traverse (x, y) or (y, x). Hence, connectivity is sufficient to guarantee loop-back connectivity in the event of an edge failure. Since Condition 2 implies that (x, y) cannot be an arc of B and R, Condition 2 ensures that loop-back connections on R do not travel over the same arc as primary connections on B, and vice-versa. Therefore, any algorithm that builds a graph B with properties 1 and 2 will suffice. The algorithm presented below is one such algorithm.

We start by choosing an arbitrary directed cycle $(c_1, \ldots, c_k, c_1)$ of G with at least three nodes ($k \geq 3$). Such a cycle is guaranteed to exist if G is edge-redundant. If this cycle does not include all nodes in the graph, we then choose a new directed path or cycle that starts and ends on the cycle and passes through at least one node not on the cycle. If the new graph does not include all nodes of the graph, we again construct another directed path or cycle, starting on some node already included, passing through one or more nodes not included, and then ending on another already included node. The algorithm continues to add new nodes in this way until all nodes are included. We now formally present the algorithm followed by the proof of its correctness.

DIRECTION SELECTION FOR RECOVERY FROM LINK FAILURES

1. Set $j = 1$.
2. Choose any cycle $(c_1, \ldots, c_k, c_1)$ in the graph with $k \leq 3$.
3. Set $B_1 = (N_1, A_1)$ where

$$N_1 = \{c_1, \ldots, c_k\}$$
$$A_1 = \{(c_1, c_2), \ldots, (c_{k-1}, c_k), (c_k, c_1)\}$$

4. If $N_j = N$, then set $B = B_j$, $R = \underline{B}$ and terminate.
5. $j := j + 1$.
6. Choose a path or cycle $\mathrm{pc}_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,L_j})$ such that $x_{j,0}, x_{j,L_j} \in N_{j-1}$ and such that the other vertices, $x_{j,i}$, $1 \leq i < L_j$ are chosen outside of N_{j-1}.

   For a path, we require $x_{j,0} \neq x_{j,L_j}$. For a cycle, we require $L_j < 3$ and $x_{j,0} = x_{j,L_j}$.
7. Set $B_j = (N_j, A_j)$ where

$$N_j = N_{j-1} \cup \{x_{j,1}, \ldots, x_{j,L_{j-1}}\}$$
$$A_j = A_{j-1} \cup \{(x_{j,0}, x_{j,1}), (x_{j,1}, x_{j,2}), \ldots, (x_{j,L_{j-1}}, x_{j,L_j})\}$$

8. Go to step 4.

We first show that the algorithm for the edge-redundant case terminates if the graph is two edge-connected. We proceed by contradiction. The algorithm would fail to terminate correctly if and only if, at step 6, no new path or cycle pc_j could be found, but a vertex in N was not included in N_{j-1}. We, therefore, assume, for the sake of contradiction, that such a vertex exists. Since the graph is connected, there is an edge $e = [x, y]$ that connects some x in N_{j-1} to some y in $N \backslash N_{j-1}$. Since the graph is edge-redundant, there exists a path between x and y that does not traverse e. Let $f = [w, z]$ be the last edge from which this path exits N_{j-1}, i.e., $w \in N_{j-1}$ and $z \in N \backslash N_{j-1}$. Note that w and x can be the same, but if $x = w$, then $z \neq y$. Similarly, y and z may be the same, but then $x \neq w$. Now, there exists a path from x to w, passing through y, which would be selected at step 6. Thus, we have a contradiction.

It is easy to see that Condition 2 is satisfied. If (x, y) is already included in the directed sub-graph, then Step 6 ensures that (y, x) cannot be added. Therefore, all that remains to be shown is that B is connected. We use induction on the sub-graphs B_j, B_1 is obviously connected. Indeed, B_1 is an unidirectional ring. Assume B_{j-1} is connected, for $j > 2$. We need to show for all $x, y \in N_j$, there is a directed path from x to y in B_j. There are four cases as follows: 1) $x, y \in N_{j-1}$; 2) $x, y \in N_j \backslash N_{j-1}$; 3) $x \in N_j \backslash N_{j-1}$, $y \in N_{j-1}$; and 4) $x \in N_{j-1}$, $y \in N_j \backslash N_{j-1}$.

Case 1 follows from the induction hypothesis and the fact that A_j is a superset of A_{j-1}.

For case 2, we have that $x, y \in \mathrm{pc}_j$. Pick l and k such that $x_{j,l} = y$ and $x_{j,k} = x$. If $l > k$, i.e., y comes after x on the path $(x_{j,1}, \ldots, x_{j,L_{j-1}})$, then $(x, x_{j,l+1}, \ldots, x_{j,k-1}, y)$ is a path from x to y in B_j. If $l < k$, i.e., y comes before x on the path $(x_{j,1}, \ldots, x_{j,L_{j-1}})$, then there exists a path from x to x_{j,L_j} on pc_j and a path from $x_{j,0}$ to y on pc_j. If $x_{j,0} = x_{j,L_j}$, then $(x, x_{j,k+1}, \ldots, x_{L_j}, x_{j,1}, \ldots, y)$ is a path from y to x in B_j. If $x_{j,0} \neq x_{j,L_j}$, then, by the induction hypothesis, there exists a path $p(x_{j,L_j}, x_{j,0})$ from x_{j,L_j} to $x_{j,0}$ in B_{j-1} and, hence, on B_j. Therefore, $(x, x_{j,k+1}, \ldots, x_{L_j}, p(x_{j,L_j}, x_{j,0}), x_{j,1}, \ldots, x)$ is a path from x to y.

For case 3, we have $x \in \mathrm{pc}_j$, $y \in N_{j-1}$. Pick k such that $x_{j,k} = x$. Now, by the induction hypothesis, there exists a path from x_{j,L_j} to y. Vertex x is, therefore, connected to y since there is a path from x to x_{j,L_j} on pc_j.

For case 4, we have that $y \in \mathrm{pc}_j$, $x \in N_{j-1}$. There is a path from x to $x_{j,0}$ by the induction hypothesis, and from $x_{j,0}$ to y on pc_j.

A very simple network management protocol will enable recovery using our choice of directions created by the above algorithm. When recovering from an arc failure on the primary (secondary) digraph, the protocol need only broadcast on the secondary (primary) digraph. Each node retains only the first copy of the broadcast and releases all unnecessary connections created by the broadcast. This simple concept is embedded in the protocol presented in Section II-C.

B. Generalized Loop-Back for Recovery From Node Failures

While the previous section dealt with recovery from a link failure, in this section, we consider a node failure. A node failure entails the failure of all links incident upon the failed node. Hence, the failure of a node requires other techniques than those used for link failure. Let us first overview the operation of loop-back in a mesh network when there is failure of a node. Each node connected by a link to the failed node, i.e., adjacent to the failed node, independently performs loop-back in the same manner as if the link connecting the failed node to the looping node had failed. We assume that only one primary connection per wavelength is incident upon each node, but that there may be several outputs along one wavelength per node. Thus, we allow the use of multicasting at nodes. The purpose of our restriction on the connections through a node is to ensure that, after loop-back, there are no collisions in the back-up graph. Multicasting applications are particularly attractive for WDM networks because splitting at optical nodes offers a simple and effective way of performing multicasting. Note that two types of traffic are looped back: traffic destined for the failed node and traffic that only traversed the failed node. Let us first consider the first type of traffic in the case where a node, say y, performs loop-back on the link between y and node x, the failed node. Node y receives on a back-up channel traffic intended for node x. Only two cases are possible: either link $[y, x]$ failed, but node x is still operational or node x failed. Note that we have made no assumption regarding the ability of the network management system to distinguish between the failure of a node and the failure of a link. Indeed, the nodes may only be aware that links have ceased to function, without knowing whether the cause is a single link failure or a node failure. Since we have a node-redundant network, our loop-back mechanism can recover from failure of node y, which entails failure of link $[y, x]$. Hence, even if there has been failure of link $[y, x]$ only, node y can eliminate all back-up traffic destined to node x because the back-up mechanism ensures that back-up traffic destined for node x arrives to node x even after failure of node y. If node x failed, then eliminating back-up traffic destined for x will prevent such back-up

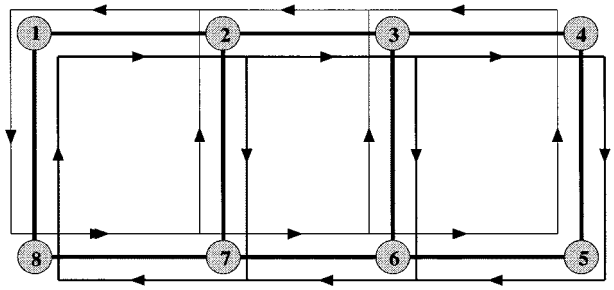


Fig. 3. Example of assignment of directions for generalized loop-back recovery from node failure.

traffic from recirculating in the network since recirculation would cause collisions and congestion. Thus, regardless of whether a node failure or a link failure occurred, back-up traffic destined for the failed node will be eliminated when a node adjacent to the failed node receives it. In SONET SHRs, a similar mechanism eliminates traffic intended for a failed node.

We may now illustrate our mechanism with a specific example applied to the network we have been considering. Fig. 3 shows a sample set of directions that can be selected for generalized loop-back recovery from node failure. Let us first consider the case where we have a primary connection along the full line from node 1 to node 3 via node 2 and node 2 fails. The shortest loop-back path is node 1 $\rightarrow$ node 8 $\rightarrow$ node 7 $\rightarrow$ node 6 $\rightarrow$ node 3. Let us now consider the case where we have a primary connection along the full line from node 1 to node 2 and node 2 fails. Then, the back-up path goes from node 8 to node 7, which eliminates the connection, because node 7 is adjacent to node 2.

We model our network as a vertex-redundant undirected graph $G = (N, E)$. We seek a directed spanning sub-graph $B = (N, A)$ of G, and its associated reversal $R = \underline{B} = (N, \underline{A})$, such that the following occurs.

- **Condition 1:** B is connected, i.e., there is a directed path in B between any two nodes.
- **Condition 2:** $(x, y) \in A \Rightarrow (y, x) \notin A$.
- **Condition 3:** For all $x, n, y \in N$ such that (x, n), (n, y) are arcs of B, there exists a directed path from x to y in R, which does not pass through n.

As in the edge-redundant case, Condition 1 ensures that any connection can be routed on B or R. However, unlike the edge-redundant case, connectivity is insufficient to guarantee loop-back connectivity after failure. Also, as in the edge-redundant case, Condition 2 ensures that loop-back connections on R do not travel over the same arc as primary connections on B, and vice-versa. Condition 3 ensures that loop-back can be successfully performed and is equivalent to the statement that all three adjacent nodes in B, x, n, y are contained in a cycle of B.

We perform loop-back for node failures in the same manner as described above for link failures. For instance, let us select two distinct nodes w and z. Let p_1 be the path in B from w to z and let n be a node other than w or z traversed by p_1. We consider the nodes x and y such that (x, n) and (n, y) are traversed in that order in p_1. Thus, (x, n), (n, y) are in A. Let p_2 be a path in R, which does not include vertex n and which goes from vertex x to vertex y. We perform loop-back from w to z using paths p_1, p_2 at node n by traversing the following circuit:

- from w to x, we use path p_1;
- at x, we loop-back from primary to secondary;
- from x to y, we use path p_2;
- at y, we loop-back from secondary to primary;
- from y to z, we use path p_1.

As discussed previously, this loop-back is more general than the type of loop-back used in a ring. In particular, the loop-back is not restricted to use a back-haul route traversing successively w, x, w, z, y, z. In order to guarantee loop-back, it is sufficient to select B and R so that, in the event of any vertex (edge) failure affecting B or R, there exists a working path around the failure on the other sub-graph.

Any sub-graph satisfying Conditions 1–3 is sufficient to perform loop-back as described above. The algorithm below guarantees these conditions by amending the algorithm for the edge-redundant case. The edge-redundant algorithm fails to insure Condition 3 for two reasons. The first reason is that cycles are allowed in Step 6, i.e., $\mathrm{pc}_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,0})$ is possible in iteration j and, hence, failure of node $x_{j,0}$ would leave both B and R disconnected. The second and more fundamental reason is that the ordering of the nodes on the added paths in steps 6 and 7 is very unrestrictive.

Our algorithm starts by choosing a directed cycle of at least three vertices containing some arbitrary edge $e = [t, s]$. If this cycle does not include all nodes in the graph, we then choose a directed path that starts on some node in the cycle, passes through some set of nodes not on the cycle, and ends on another node on the cycle. If the cycle and path above do not include all vertices of the graph, we again construct another directed path, starting on some node already included, passing through one or more nodes not included, and then ending on another already included node. The algorithm continues to add new nodes in this way until all nodes are included.

It is simple to show that, in a vertex-redundant graph, for any edge e, a cycle with three vertices must exist containing e. It can also be seen that, for any such cycle, a path can be added as above, and subsequent paths can be added, in arbitrary ways, until all nodes are included. It is less simple to choose the direction of the added paths and, hence, the B and R directed sub-graphs. The technique we present relies in part on results presented [6], [19], [21], and [22]. We now present the algorithm followed by the proof of its correctness.

DIRECTION SELECTION FOR RECOVERY FROM NODE FAILURES

1. Set $j = 1$. Pick an arbitrary edge $e = [s, t]$ and assign $v(s) = V > 0$ and $v(t) = 0$.
2. (a) Choose any cycle $(s, c_1, \ldots, c_{k-1}, t, s)$ in the graph with $k \geq 2$.
   (b) Order these nodes by assigning values such that $v(s) = V > v(c_1) > \cdots > v(c_{k-1}) > v(t) = 0$.

3. Set $B_1 = (N_1, A_1)$ where

$$N_1 = s, c_1, \ldots, c_{k-1}, t$$
$$A_1 = (s, c_1), (c_1, c_2), \ldots, (c_{k-1}, c_k), (c_k, t), (t, s)$$

4. If $N_j = N$, then set $B = B_j$, $R = \underline{B}$ and terminate.
5. $j := j + 1$.
6. (a) Choose a path $p_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,L_j})$, $L_j > 2$, in the graph such that

$x_{j,0}, \ldots, x_{j,L_j} \in N_{j-1}$, with $v(x_{j,0}) > v(x_{j,L_j})$. The other vertices, $x_{j,i}$, $1 \leq i < L_j$,

are chosen outside of N_{j-1}.

(b) Order the new vertices by assigning values such that $v(x_{j,0}) > v(x_{j,1}) > \cdots > v(x_{j,L_{j-1}}) > v_{\max}$ where

$$v_{\max} = \max_{y \in N_{j-1}} \{v(y) : v(y) < v(x_{j,0})\}.$$

7. Set $B_j = (N_j, A_j)$ where

$$N_j = N_{j-1} \cup \{x_{j,1}, \ldots, x_{j,L_{j-1}}\}$$
$$A_j = A_{j-1} \cup \{(x_{j,0}, x_{j,1}), (x_{j,1}, x_{j,2}), \ldots, (x_{j,L_{j-1}}, x_{j,L_j})\}.$$

8. Go to step 4.

Note in step 6b that $v_{\max} \geq v(x_{j,L_j})$. We first show that the algorithm for the node-redundant case terminates if the graph is vertex-redundant. We shall proceed by contradiction. The algorithm would fail to terminate correctly if and only if at step 6 no new path p_j could be found, but a vertex in N was not included in N_{j-1}. We assume for the sake of contradiction that such a vertex exists. Since the graph is connected, there is an edge $[x, y]$, which connects some x in N_{j-1} to some y in $N \backslash N_{j-1}$. Pick a vertex $q \in N_{j-1}$, such that $q \neq x$. Since the graph is node-redundant, there exists a path between y and q, which does not use x. Let $f = [w, z]$ be the last edge from which this path exits N_{j-1}, i.e., $w \in N_{j-1}$ and $z \in N \backslash N_{j-1}$. Note that $w = q$ or $y = z$ is possible. Now there exists a path from x to w, passing through y, which would be selected at step 6 in the algorithm. Therefore, we have a contradiction.

We now prove that B satisfies Conditions 1–3. The fact that B is connected follows by induction on j using almost identical arguments as used in the proof for the link-redundant case. In particular, we can see by induction on j that there is a directed path in B_j from $x \in N_j$ to any $y \in N_j$. Since these properties hold for each j, they also hold for the final directed sub-graph B. We may, therefore, state that B is connected. As in the edge-redundant case, Condition 2 is satisfied by the restrictions on adding new arcs.

Finally, we prove that B satisfies Condition 3. We need to prove the fact that, for all $x, n, y \in N$ such that (x, n), (n, y) are arcs of B, there exists a directed path from x to y in R, which does not pass through n. Since R is the reversal of B, we can prove the equivalent statement that there exists a directed path from y to x in B, which does not pass through n. The key observation is to note that arc $(t, s) \in A$ has a special property. In particular, it is the only arc in B for which the value of the originating node is lower than the value of the terminating node, i.e., $v(t) = 0 < v(s) = V$. Thus, for all $(i, j) \in A$, $v(i) > v(j)$, unless $i = t$ and $j = s$. From this property, it immediately follows that all directed cycles in B contain (t, s). To see this, let $x_0, x_1, \ldots, x_k, x_0$ be a cycle and note that, if (t, s) were not traversed in this cycle, then $v(x_0) > v(x_1) > \cdots > v(x_k)$ and, hence, (x_k, x_0) could not be an arc in B. Also, since B is connected, we also have that (t, s) is the unique arc into s in B for, otherwise, we could construct a cycle through s, which did not pass through t.

Only two cases need to be considered to prove the desired property $n = s$ and $n \neq s$. First consider $n = s$. Since B is connected, there exists a path from y to x in B and this path need not include s since the only way to reach s is through $x = t$. Now consider $n \neq s$. There exists paths $p(y, n) = (y, y_1, \ldots, y_k, n)$ from y to n and $p(n, x) = (n, x_m, x_{m-1}, \ldots, x_1, x)$ from n to x, both in B. Since $(y, y_1, \ldots, y_k, n, y)$ is a cycle, it includes s. Similarly, $(n, x_m, x_{m-1}, \ldots, x_1, x, n)$ is a cycle and, hence, includes s. Therefore, there is a path starting at y, proceeding on $p(y, n)$ until s (which is before n in $p(y, n)$), starting in $p(n, x)$ at s (which is after n in $p(n, x)$), and ending at x.

C. Protocol

We now overview a protocol that ensures proper recovery, using generalized loopback for node or link recovery. Our protocol is more involved than that needed to recover only from a link failure, since we must contend with the failure of all links adjacent to the failed node. However, our algorithm will also operate properly for link failures without node failures. Our protocol uses negative acknowledgment (NACK) and labels to establish correct rerouting. The signaling for the protocol may be performed over an out-of-band control channel or an in-band control channel, such as a subcarrier multiplexed signal.

Consider the failure of a primary fiber from x to y. Failure of the fiber may be due to failure of the fiber itself or of node y. When x detects the failure, it writes "y" into the failure label and loops the primary stream back into the back-up digraph, splitting it across all outgoing arcs in the back-up digraph. As the traffic enters each new node, the node forwards the traffic, again splitting it over all outgoing arcs. Backup fibers leaving a node can be preconfigured to split an incoming stream, shortening the time required to flood failure information across outgoing links. For nodes with only one incoming stream, the route is fully preplanned, and no traffic is lost during the decision process. For nodes with more than one incoming stream, the first of the streams to receive traffic is chosen for forwarding. A stream that becomes active after the first—typically owing to traffic from the same failure arriving via a different route—is dropped, and a NACK is returned on the reverse back-up arc. A node that receives a NACK on an outgoing link ceases to forward traffic on that link. If all outgoing links for a node are NACKed, the node propagates a NACK on its own incoming link, in effect releasing the connection on that link. If all outgoing links at x are NACKed, recovery has failed (possibly multifailure scenarios or scenarios where several connections over the same wavelength were present at a failed node).

The NACK-based protocol can be extended with hop-count and signal-power (splitting) restrictions to reduce the area over which a failure propagates, but such restrictions require more careful selection of the back-up digraph to guarantee recovery

from all single failures and to prevent significant degradation of multifailure recovery possibilities.

The use of NACKs serves to limit the use of back-up arcs to those necessary to recovery. Another approach to achieving this goal is to mark the successful route and forward tear-down messages down all other arcs. The NACK scheme is superior to this approach in two ways. First, tear-down messages must catch up with the leading edge of the traffic, but cannot travel any faster. In the worst case, a route is torn down only to allow a cyclic route to recreate itself, resulting in long-term traffic instabilities in the back-up digraph. To avoid this possibility, tear-down requires that nodes remember the existence of failure traffic between the time that they tear down the route and the time that the network settles (a global phenomenon). A second point in favor of the NACK-based scheme is that it handles multicast (possibly important for node failures) naturally by discarding only unused routes. A tear-down scheme must know the number of routes to recover in advance or discover it dynamically; the first option requires a node to know the routes through its downstream neighbors, while the second option is hard because of timing issues (when have all routes been recovered?).

Meanwhile, y detects the loss of the stream and begins listening for traffic with its name or x's name on the back-up stream. The second case handles failure of node x, which results in flooding of traffic destined for x. Note that traffic for x can also be generated owing to failure of a primary arc ending at x, but, in such a case, y does not detect any failure and does not listen for back-up traffic. Once a stream with either name is received, it is spliced into the primary traffic stream, completing recovery. Other paths are torn down through NACKs. Note that if a stream ends at a failed node x, no node listens for the back-up traffic for node x, and all connections carrying back-up traffic for node x are eventually torn down.

While our protocol for node failure is more complicated than that for link failure, it is still relatively simple. Node failure in ring-based systems is a very complex operation whenever a node is on more than one ring. For double-cycle cover, node recovery requires hopping among rings and, thus, necessitates a centralized controller with global knowledge of the network. Even for simple double-homed SONET rings, node recovery involves the use of matched nodes. Proper operation of matched nodes requires significant inter-ring signaling as well as dual-fed path protection between rings.

III. Applications

A. Choice of Routings

In this section, we present results from heuristic algorithms for selecting directions in the back-up graph. We seek to select directions in such a way to avoid excessive length for back-up paths. We consider three different algorithms. The first algorithm, which we term Heuristic 1, first finds, for each link, a shortest loop, which includes that link. A loop is a directed cycle, thus a shortest loop is a directed cycle with the minimum number of links. We order the shortest loops of all the links in ascending order of length. Shortest loops with equal lengths are ordered arbitrarily with respect to each other. Beginning with the first shortest loop, in ascending order, we assign, whenever possible, directions according to the directions of the arcs along the shortest loop. The second algorithm, i.e., Heuristic 2, also relies on considering shortest loops, but takes into account the fact that the choice of direction on a link may affect other links. We create a heuristic measure of this effect, which we call the associate number (AN). The AN of link is the number of different shortest loops that pass through that link. In particular, the AN can help us distinguish among links with equal length shortest loops. We order the links in ascending order of AN. We begin, as for Heuristic 1, by finding, for each link, a shortest loop, which includes that link. Links with equal ANs are ordered arbitrarily with respect to each other. Beginning with the first link and progressing in ascending order, we assign directions, whenever possible, according to the shortest loop of the link being considered. The last algorithm we consider is a random assignment of directions. While the number of possible directions is exponential in the number of links, we significantly reduce that number by requiring the directions to be feasible.

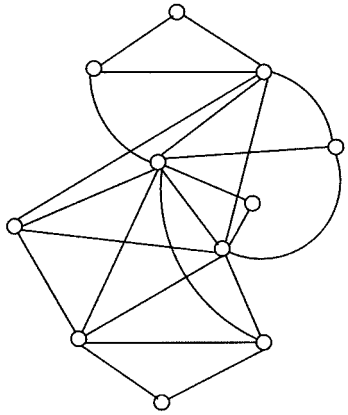


Fig. 4. NJ LATA network.

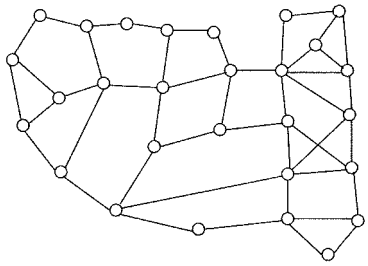


Fig. 5. LATAX network.

We apply our algorithms to three networks, NJ LATA, LATAX, and ARPANET, shown in Figs. 4–6. We consider the maximum length of a back-up path and the average length. Table I shows the results obtained from running the different algorithms for the three networks we consider. Heuristic 1 was run ten times for each network and the best result was kept for the maximum and average. Note that the same choice of directions did not always yield both the best maximum and the best average. Heuristic 2 was run in the same way as Heuristic 1. For the random algorithm, we limited ourselves to 52 runs for NJ LATA, 128 runs for LATA X, and 123 runs for ARPANET. The best maximum and the best average were chosen in each case. Comparing the running time of running the

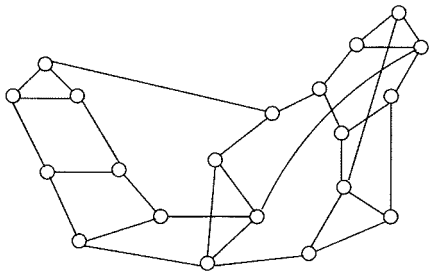


Fig. 6. ARPANET network.

TABLE I
COMPARISON OF THE BEST RESULTS BETWEEN HEURISTIC ALGORITHMS AND THE METHOD OF SELECTING DIRECTIONS RANDOMLY

	Random		Heuristic 1		Heuristic 2	
	Max.	Avg.	Max.	Avg.	Max.	Avg.
NJLATA	3	2.086957	2	2.00000	2	2.00000
LATAX	9	3.543478	4	2.95652	5	2.86957
ARPANET	9	4.093750	8	3.81250	8	3.35484

heuristic algorithms ten times versus the above number of times for the random algorithm, we obtain that Heuristic 1 yielded a run time improvement of 72%, 90%, and 88% over a random choice of directions for NJ LATA, LATAX, and ARPANET, respectively. Heuristic 2 yielded a run time improvement of 73%, 91%, and 90% over random choice of directions for NJ LATA, LATAX, and ARPANET, respectively.

B. WDM-Based Loop-Back Recovery

In fiber-based restoration, the entire traffic carried by a fiber is backed by another fiber. In fiber-based restoration, it does not matter whether the system is a WDM system. If traffic is allowed in both directions in a network, fiber-based restoration relies on four fibers, as illustrated in Fig. 7. In WDM-based recovery, restoration is performed in a wavelength-by-wavelength basis.

WDM-based recovery requires only two fibers, although it is applicable to a higher number of fibers. Fig. 8 illustrates WDM-based recovery. A two-fiber counter-propagating WDM system can be used for WDM-based restoration, even if traffic is allowed in both directions. Note that WDM restoration, as shown on Fig. 8, does not require any change of wavelength. Thus, traffic initially carried by λ_1 is backed up by the same wavelength. Obviating the need for wavelength changing is economical and efficient in WDM networks. One could, of course, back up traffic from λ_1 on fiber 1 onto λ_2 on fiber 2, if there were advantages to such wavelength changing, for instance, in terms of wavelength assignment for certain traffic patterns. We can easily extend the model to a system with more fibers, as long as the back-up for a certain wavelength on a certain fiber is provided by some wavelength on another fiber. Moreover, we may change the fiber and/or wavelengths from one fiber section to another. For instance, the back-up to λ_1 on fiber 1 may be λ_1 on fiber 2 on a two-fiber section and λ_2 on fiber 3 on another section with four fibers. Note also, that we could elect not to back up λ_1 on fiber 1 and instead use λ_1 on fiber 1 for primary traffic. The extension to systems with more fibers, inter-wavelength back-ups and back-ups among fiber sections can be readily done.

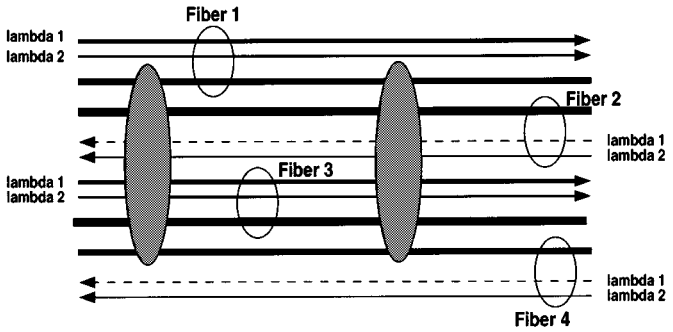


Fig. 7. Four-fiber system with fiber-based loop-back. Primary traffic is carried by fiber 1 and by fiber 2. Backup is provided by fiber 3 for fiber 1 and by fiber 4 for fiber 2.

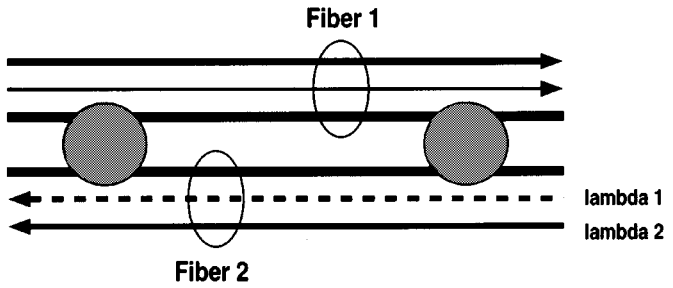


Fig. 8. Two-fiber WDM-based loop-back. Primary traffic is carried by fiber 1 on λ_1 and by fiber 2 on λ_2. Backup is provided by λ_1 on fiber 2 for λ_1 on fiber 1. λ_2 on fiber 2 is backed up by λ_2 on fiber 1.

There are several advantages to WDM-based recovery systems over fiber-based systems. The first advantage is that, if fibers are loaded with traffic at one-half of total capacity or less, then only two fibers rather than four are needed to provide recovery. Thus, a user need only lease two fibers, rather than paying for unused bandwidth over four fibers. On existing four-fiber systems, fibers could be leased by pairs rather than fours, allowing two leases of two fibers each for a single four-fiber system. The second advantage is that, in fiber-based systems, certain wavelengths may be selectively given restoration capability. For instance, one-half the wavelengths on a fiber may be assigned protection, while the rest may have no protection. Different wavelengths may thus afford different levels of restoration QoS, which can be reflected in pricing. In fiber-based restoration, all the traffic carried by a fiber is restored via another fiber. If each fiber is less than one-half full, WDM-based loop-back can help avoid the use of counterpropagating wavelengths on the same fiber. Counterpropagating on the same fiber is onerous and reduces the number of wavelengths that a fiber can carry with respect to unidirectional propagation. Our WDM-based loop-back may make using two unidirectional fibers preferable to using two counterpropagating fibers.

We may now draw a comparison between generalized loop-back and double-cycle covers for WDM-based loop-back recovery. The ability to perform restoration over mesh topologies was first introduced in [3] and [4]. In particular, [4] considers link failure restoration in optical networks with arbitrary two-link redundant arbitrary mesh topologies and bi-directional links. The scheme relies on applying methods for double-cycle covers to restoration.

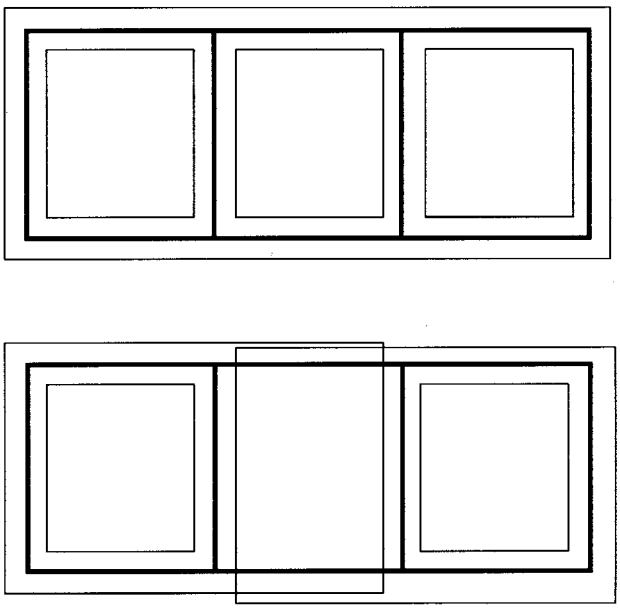


Fig. 9. Two possible double-cycle covers for a sample network.

Let us first discuss how double-cycle ring covers can be used to perform recovery. A double-cycle ring cover covers a graph with cycles in such a way that each edge is covered by two cycles. Cycles can then be used as rings to perform restoration. Each cycle corresponds either to a primary or a secondary two-fiber ring.

Let us consider a link covered by two rings, rings 1 and 2. If we assign a direction to ring 1 and the opposite direction to ring 2, then ring-based recovery using the double-cycle cover uses ring 2 to back up ring 1. In effect, this recovery is similar to recovery in conventional SHRs, except that the two rings that form four-fiber SHRs are no longer co-located over their entire length. Fig. 9 shows the two possible double-cycle covers, shown in thin lines, for a certain fiber topology, shown in bold lines. In the case of four fiber systems, with two fibers in the same direction per ring, we have fiber-based recovery, because fibers are backed up by fibers. For the type of WDM-based loop-back we consider in this section, each ring is both primary for certain wavelengths and secondary for the remaining wavelengths. For simplicity, let us again consider just two wavelengths. Figs. 10 and 11 show that we cannot use one ring to provide WDM-based loop-back back-up for another unless we perform wavelength changing. We cannot assign primary and secondary wavelengths in such a way that a wavelength is secondary or primary over a whole ring.

We may point out another drawback of the back-up paths afforded by double-cycle covers. In Fig. 10, a break on a link may cause one direction to be backed up on ring 1, while another direction may be backed up on ring 4. The two directions on a link will, therefore, have different delays in their restoration time and incur different timing jitters. Such asymmetry does not occur in SHRs or in generalized loop-back since the back-up paths for both directions traverse the same links.

C. Plurality of Back-Up Routes for Generalized Loop-Back

We have mentioned that our algorithm can be used to perform recovery even when there is a change in the conditions of the networks. We give a brief example, shown in Fig. 12, of how such flexibility is afforded. We show a recovery sub-graph built for link failure restoration. For the case of failure of the link between node 2 and node 3, a curved line shows the recovery back-up path. The shortest back-up path uses the link between

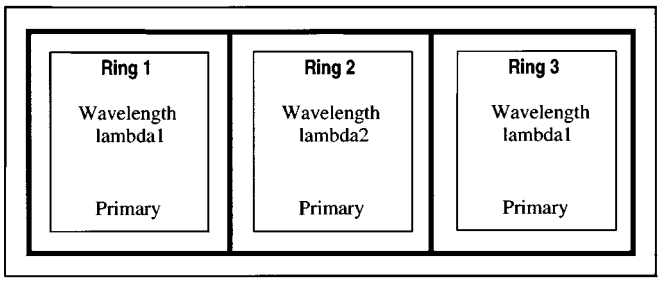


Fig. 10. Illustration of the fact that using the ring cover derived from the first double-cycle cover in Fig. 9 does not yield two-fiber WDM loop-back.

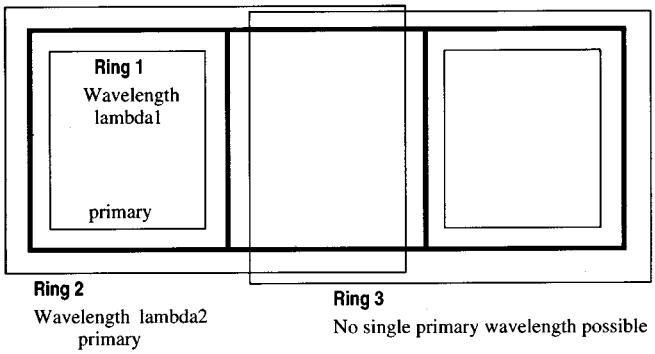
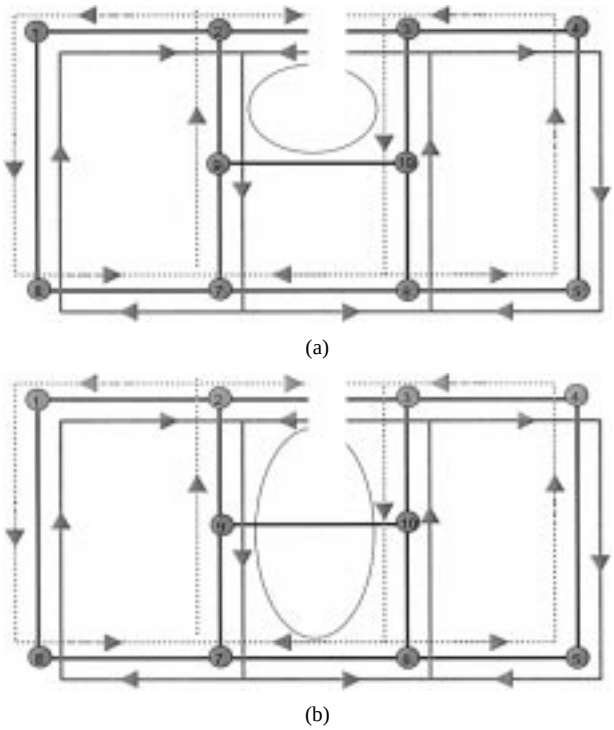


Fig. 11. Illustration of the fact that using the ring cover derived from the second double-cycle cover in Fig. 9 does not yield two-fiber WDM loop-back.



(a)

(b)

Fig. 12. Example of two different back-up paths.

nodes 9 and 10, as shown on Fig. 12. Suppose the link between nodes 9 and 10 becomes unusable for the back-up path, for instance because the link has failed or because all wavelengths on that link are used for extra traffic. Then, the back-up path for a failure between nodes 2 and 3 can be the path node 3 $\rightarrow$ node 10 $\rightarrow$ node 6 $\rightarrow$ node 7 $\rightarrow$ node 9 $\rightarrow$ node 2, shown by a curved line on Fig. 12(b). Thus, the same back-up sub-graph can be used both when the link between nodes 9 and 10 is available and when it is not available.

Not all links may become unavailable. If the link between nodes 3 and 10 becomes unavailable, restoration after failure of the link between nodes 2 and 3 is not possible. However, it is possible to determine whether certain links are necessary for recovery from failure of other links. Since there are two paths in the back-up sub-graph from node 10 to node 9, the link between node 9 and node 10 is not necessary and that link can be freed up to carry extra traffic if the need arises.

IV. Summary and Conclusions

We have presented generalized loop-back, a novel way of implementing loop-back on mesh networks without using ring-based schemes. We have established routings and protocols to ensure recovery after a link or a node failure. Our method is guaranteed to work in polynomial time regardless of whether the graph representation of the network is planar or not, whereas double-cycle covers have polynomial-time solution only for planar graphs. The gist of our method is to assign directions to fibers and the wavelengths traveling through them. Our method allows great flexibility in planning the configuration of a network, as long as it has redundancy, while providing the bandwidth utilization advantages typically associated with loop-back protection in SHRs. Recovery, as in SONET BLSR, is performed by the nodes adjacent to the link or node failure. Moreover, our loop-back recovery method does not require the nodes performing loop-back to distinguish between a node and link failure. We have shown that simple heuristic algorithms yield satisfactory results in terms of average and maximum length of back-up paths. We have compared our method to the previously known method for loop-back for link failure on mesh networks. That method [4] is based upon double-cycle covers and we have shown that such a method may not be applied to WDM-based loop-back systems. Moreover, we have shown by a simple example that generalized loop-back allows recovery to be performed in a bandwidth-efficient manner.

There are several areas of further work. One of them is considering the issue of wavelength assignment jointly with back-up considerations, whether the back-up be loop-back, APS, or hybrid. Another issue is the determination of the back-up path. Broadcasting, or flooding, in the back-up wavelength causes that wavelength to be unavailable in parts of the network that are not required to provide back-up. Some methods for choosing back-up paths are presented in [7].

Another area for further research is the use of generalized loop-back to perform bandwidth-efficient recovery. As we discussed in Section II-A, link and node restoration generally are less efficient, in terms of capacity utilization, than event-triggered path restoration. However, our scheme allows recovery of links that are not included in the back-up sub-graph, as long as the end nodes are included in the back-up sub-graph. This operation can be viewed as being akin to p-cycles, but with greater flexibility in the choice of the back-up sub-graph. Eliminating links from the back-up sub-graph is economical in bandwidth, but entails some degradation in terms of performance metrics, such as length of back-up path or recovery from two failures. Recent results [23] have shown that significant savings, of the order of 25%, can be achieved using generalized loop-back over several networks without sacrificing the length of the longest back-up and the ability to recover from double failures.

Acknowledgment

The authors thank the reviewers and P. Lin for their comments.

References

[1] G. N. Brown, W. D. Grover, J. B. Slevinsky, and M. H. MacGregor, "An architecture for efficient survivable networks," in *Proc. IEEE GLOBECOM*, vol. 2, 1994, pp. 471–477.

[2] R. Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques," *J. Network Syst. Manag.*, vol. 2, 1994.

[3] G. Ellinas and T. E. Stern, "Automatic protection switching for link failures in optical networks with bi-directional links," in *Proc. IEEE GLOBECOM*, vol. 1, 1996, pp. 152–156.

[4] G. Ellinas, R. G. Hailemarian, and T. E. Stern, "Protection cycles in mesh WDM networks," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1924–1937, Oct. 2000.

[5] G. Fan, "Covering graphs by cycles," *SIAM J. Comput.*, vol. 5, pp. 491–496, Nov. 1992.

[6] S. G. Finn, M. Médard, and R. A. Barry, "A novel approach to automatic protection switching using trees," presented at the Proc. Int. Conf. Commun., 1997.

[7] ——, "A new algorithm for bi-directional self healing for arbitrary redundant networks," presented at the Proc. Opt. Fiber Commun. Conf., 1998.

[8] I. Fournier, "Longest cycles in 2-connected graphs of independence number α," in *Cycles in Graphs, Annals of Discrete Mathematics*. Amsterdam, The Netherlands: North-Holland, 1985, vol. 115, pp. 201–204.

[9] L. Goddyn, "A girth requirement for the double cycle cover conjecture," in *Cycles in Graphs, Annals of Discrete Mathematics*. Amsterdam, The Netherlands: North-Holland, 1985, vol. 115, pp. 13–26.

[10] W. D. Grover, "Case studies of survivable ring, mesh and mesh-arc hybrid networks," in *Proc. IEEE GLOBECOM*, 1992, pp. 633–638.

[11] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network reconfiguration," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, 1998, pp. 537–543.

[12] R. Häggkvist and B. Jackson, "A note on maximal cycles in 2-connected graphs," in *Cycles in Graphs, Annals of Discrete Mathematics*. Amsterdam, The Netherlands: North-Holland, 1985, vol. 115, pp. 205–208.

[13] A. Itai, R. J. Lipton, C. H. Papadimitriou, and M. Rodeh, "Covering graphs with simple circuits," *SIAM J. Comput.*, vol. 10, pp. 746–750, 1981.

[14] B. Jackson, "Hamilton cycles in regular 2-connected graphs," *J. Comb. Theory*, ser. B, vol. 29, pp. 27–46, 1980.

[15] F. Jaeger, "A survey of the double cycle cover conjecture," in *Cycles in Graphs, Annals of Discrete Mathematics*. Amsterdam, The Netherlands: North-Holland, 1985, vol. 115, pp. 1–12.

[16] S. Dravida, J. Anderson, B. T. Doshi, and P. Harshavardhana, "Fast restoration of ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 128–136, Jan. 1994.

[17] R.-H. Jan, F.-J. Hwang, and S.-T. Cheng, "Topologicial optimization of a communication network subject to a reliability constraint," *IEEE Trans. Rel.*, vol. 42, pp. 63–70, Mar. 1993.

[18] S. Venkatesan, J. Veerasamy, and J. C. Shah, "Spare capacity assignment in telecom networks using path restoration," in *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 370–374.

[19] M. Médard, S. G. Finn, and R. A. Barry, "Automatic protection switching for multicasting in optical mesh networks," in *Proc. Opt. Fiber Commun. Conf.*, 1997.

[20] ——, "WDM loop-back recovery in mesh networks," in *Proc. IEEE INFOCOM*, vol. 2, 1999, pp. 752–759.
[21] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Networking*, vol. 7, pp. 641–652, Oct. 1999.
[22] M. Médard, S. G. Finn, R. G. Gallager, and R. A. Barry, "Redundant trees for automatic protection switching in arbitrary node-redundant or edge-redundant graphs," presented at the Proc. ICC, 1998.
[23] M. Médard, S. S. Lumetta, and Y. C. Tseng, "Capacity-efficient restoration for optical networks," in *Proc. Opt. Fiber Commun. Conf.*, 2000, pp. 207–209.
[24] K. T. Newport and P. K. Varshney, "Design of survivable communication networks under performance constraints," *IEEE Trans. Rel.*, vol. 40, pp. 433–440, Oct. 1991.
[25] M. H. MacGregor, R. R. Iraschko, and W. D. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 325–336, June 1988.
[26] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks, part I—Protection," in *Proc. IEEE INFOCOM*, 1999, pp. 744–751.
[27] P. D. Seymour, "Sums of circuits," in *Graph Theory and Related Topics*, U. S. R. Murty and J. A. Bondy, Eds. New York: Academic, 1979, pp. 341–355.
[28] M. Stoer, *Design of Survivable Networks*. Berlin, Germany: Springer-Verlag, 1992.
[29] G. Szekeres, "Polyhedral decomposition of cubic graphs," *J. Australian Math. Soc.*, vol. 8, pp. 367–387, 1973.
[30] C. Thomassen, "On the complexity of finding a minimum cycle cover of a graph," *SIAM J.*, vol. 26, pp. 675–677, June 1997.
[31] O. J. Wasem, "An algorithm for designing rings for survivable fiber networks," *IEEE Trans. Rel.*, vol. 40, pp. 428–432, Oct. 1991.
[32] T. H. Wu, R. H. Caldwell, and M. Boyden, "A multi-period design model for survivable network architecture selection for SDH/SONET interoffice networks," *IEEE Trans. Rel.*, vol. 40, pp. 417–432, Oct. 1991.
[33] T. H. Wu and S. F. Habiby, "Strategies and technologies for planning a cost-effective survivable network architecture using optical switches," *IEEE Trans. Rel.*, vol. 8, pp. 152–159, Feb. 1991.
[34] T. H. Wu, D. J. Kolar, and R. H. Cardwell, "Survivable network architectures for broad-band fiber optic networks: Model and performance comparison," *IEEE J. Lightwave Commun.*, vol. 6, pp. 1698–1709, Nov. 1988.
[35] ——, "High-speed self-healing ring architectures for future interoffice networks," in *Proc. IEEE GLOBECOM*, vol. 2, 1989, pp. 23.1.1–23.1.7.
[36] C. S. Wu, S. W. Lee, and Y. T. Hou, "Backup vp preplanning strategies for survivable multicast ATM," in *Proc. IEEE GLOBECOM*, 1997, pp. 267–271.
[37] D. R. Woodall, "Maximal circuits of graphs II," *Studia Sci. Math. Hungarian*, vol. 10, pp. 103–109, 1975.
[38] T. H. Wu, *Fiber Network Service Survivability*: Artech House, 1992.
[39] ——, "A passive protected self-healing mesh network architecture and applications," *IEEE/ACM Trans. Networking*, vol. 2, pp. 40–52, Feb. 1994.
[40] Y. Xiong and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," *IEEE J. Lightwave Commun.*, vol. 7, pp. 98–110, Feb. 1999.
[41] Y. Zhu, Z. Liu, and Z. Yu, "An improvement of Jackson's result on Hamilton cycles in 2-connected graphs," in *Cycles in Graphs, Annals of Discrete Mathematics*. Amsterdam, The Netherlands: North-Holland, 1985, vol. 115, pp. 237–247.

Muriel Médard (S'91–M'95) received the B.S. degree in electrical engineering and computer science and in mathematics, the B.S. degree in humanities, the M.S. degree in electrical engineering, and the Sc.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1989, 1990, 1991, and 1995, respectively.

She is currently an Assistant Professor in the Electrical Engineering and Computer Science Department, MIT, and is a member of the Laboratory for Information and Decision Systems. She was previously an Assistant Professor with the Electrical and Computer Engineering Department and a member of the Coordinated Science Laboratory with the University of Illinois Urbana-Champaign. From 1995 to 1998, she was a Staff Member of the Optical Communications and the Advanced Networking Groups with the MIT Lincoln Laboratory. Her research interests are in the areas of reliable communications, particularly for optical and wireless networks.

Dr. Médard was the recipient of the 2002 IEEE Leon Kirchmayer Paper Award for best paper by an author under 30.

Richard A. Barry (M'00) received the B.S., M.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge.

He is the co-founder and Chief Technology Officer of Sycamore Networks Inc., Chelmsford, MA, where he is responsible for network and product architecture. He was previously with the MIT Lincoln Laboratory, where he was the architect and co-developer of high-speed WDM and OTDM (soliton) optical networks. His experience in optical technology also includes a key role in the development of the first all-optical WDM network testbed, known as the all-optical network (AON), for which he was a principle architect. He was an Assistant Professor in the Electrical Engineering and Computer Science Department, George Washington University. He is the Editor-in-Chief of the *Journal of Optical Networking*.

Dr. Barry serves on the Board of Directors of the OIF and is also the former chairman of the OIF Architecture Working Group. He was an editor of the *IEEE Network Magazine*.

Steven G. Finn (S'70–M'75) was born in Boston, MA, in 1946. He received the B.S., M.S., and Sc.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, in 1969, 1969, and 1975, respectively, all in electrical engineering.

From 1975 to 1980, he was with the Codex/Motorola Corporation, where he held various R&D positions, including Director of Network Product Development and Director of Network Research. In 1980, he founded the Bytex Corporation, a data communications equipment manufacturer, where, from 1987 to 1990, he was the CEO and Chairman of the Board of Directors. In 1990, he rejoined MIT as a Vinton Hayes Fellow and Visiting Scientist in the Laboratory of Information and Decision Sciences. He is currently a Principal Research Scientist and Lecturer in the Department of Electrical Engineering and Computer Science, MIT, and a Senior Member of the Technical Staff at the MIT Lincoln Laboratory. His current research interests are in the areas of optical networks, high-speed data network transport, network architecture, and network management.

Wenbo He received the M.S. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, in 2001, the M.E. degree in automation from Tsinghua University, Beijing, China, in 1998, the B.E. degree in automatic control from the Harbin Engineering University, Heilongjiang, China, in 1995, and is currently working part-time toward the Ph.D. degree in computer science at the University of Illinois at Urbana-Champaign.

She is currently a Software Engineer with Cisco Systems Inc., Champaign, IL.

Steven S. Lumetta (S'97–M'98) received the A.B. degree in physics, the M.S. degree in computer science, and the Ph.D. degree in computer science from the University of California at Berkeley in 1991, 1994, and 1998, respectively.

He is currently an Assistant Professor of electrical and computer engineering and a Research Assistant Professor in the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. He has been involved in a wide range of problems in scalable parallel computing, including languages (split-C), tools (Mantis debugger), algorithms, and runtime systems, culminating in his dissertation on multiprotocol user-level communication on clusters of SMPs. His research interests are in optical networking, high-performance networking and computing, hierarchical systems, and parallel runtime software.

Dr. Lumetta is a member of the Association for Computing Machinery (ACM).