On Double-Link Failure Recovery in WDM Optical Networks

Hongsik Choi, Suresh Subramaniam, and Hyeong-Ah Choi

Abstract— **Network survivability is a crucial requirement in high-speed optical networks. Typical approaches of providing survivability have considered the failure of a single component such as a link or a node. In this paper, we consider a failure model in which any two links in the network may fail in an arbitrary order. Three loopback methods of recovering from double-link failures are presented. The first two methods require the identification of the failed links, while the third one does not. However, precomputing the backup paths for the third method is more difficult than for the first two.**

A heuristic algorithm that pre-computes backup paths for links is presented. Numerical results comparing the performance of our algorithm with other approaches suggests that it is possible to achieve 100% recovery from double-link failures with a modest increase in backup capacity.

Index Terms—**Wavelength division multiplexing (WDM), loopback recovery, restoration, double-link failures, 3-edge-connected graph.**

I. Introduction

THE explosive growth of the Internet has fueled intensive research on high-speed optical networks based on wavelength division multiplexing (WDM) technology. WDM technology harnesses the large bandwidth of the optical fiber, which is of the order of several Terabits/s into a few tens of wavelengths, each of which can be operated at electronic rates of a few Gb/s. Point-to-point WDM links with several tens of wavelengths have been deployed by carrier networks.

Recent advances in optical routing and switching are enabling the transition from point-to-point optical WDM links to true optical networking by performing optical routing. In this technique called as wavelength routing, wavelengths can be independently routed from an input port to an output port. With the potential development of optical switches that are capable of dynamically reconfiguring the routing pattern under electronic control, the flexibility provided by the network is dramatically increased. In a dynamically configurable wavelength-routing network, *lightpaths* or all-optical circuit-switched paths can be provided on a demand basis, depending on traffic requirements.

As wavelength-routing paves the way for network throughputs of possibly hundreds of Tb/s, network survivability assumes critical importance. A short network outage can lead to data losses of the order of several gigabits. Hence, protection or dedicating spare resources in anticipation of faults, and rapid restoration of traffic upon detection of a fault are becoming increasingly important. According to [1], the overall availability requirements are of the order of 99.999 percent or higher. Survivability is the ability of the network to withstand equipment and link failures. There are several kinds of failures that can disrupt the lightpath service provided by the optical network. Link failures occur because of fiber cuts that are mostly due to backhoe accidents. Such cuts typically result in all fibers in the bundle to be cut and hence a link failure could lead the failure of hundreds of channels, and therefore need elaborate restoration mechanisms. Node failures occur due to the failure of equipment at nodes such as switches. These equipment are typically protected within the node by redundant equipment (including redundant switches) [2]. The other type of failure is a channel failure in which the equipment on a single wavelength channel such as transmitter or receiver fails. These failures cause a single lightpath to fail and typically do not affect the other lightpaths. In order to recover from channel failures, spare transmitting and receiving equipment must be available at the source and destination nodes.

In this paper, we restrict ourselves to the case of link failures. Traditional optical networks have tended to take the form of point-to-point WDM transmission links or rings. Well-known protection mechanisms such as $1 + 1$ and $1 : 1$ protection [3] are used for point-to-point links. There are two kinds of protection schemes for other networks: link protection and path protection. In link protection, an alternate lightpath (called as a *backup lightpath*) between the end points of each link is *pre-computed*. Upon the link's failure, all the lightpaths using the link (called as *working lightpaths*) are switched at the end-nodes of the link to their corresponding backup lightpaths. The portion of the working lightpaths excluding the failed link remains the same. In contrast, path protection entails the rerouting of all working lightpaths that use the failed link along pre-computed backup lightpaths. Here, the entire route of the working lightpaths may be changed. This flexibility in path protection could lead to lower protection capacity but requires that all failed paths effect their recovery independently. On the other hand, link protection may require more protection capacity because of reduced flexibility in rerouting, but requires only local knowledge around the failed link to complete the recovery. In both link and path protection, the protection capacity may be dedicated to a link or path, respectively, or may be shared.

Rings (with links in both the clockwise and counterclockwise directions) have been especially attractive because of the availability of exactly one backup path between any two nodes, leading to simple automatic protection switching mechanisms. When a link fails, in link protection, the end-nodes of the link

H. Choi and H.-A. Choi are with the Dept. of Computer Science, and S. Subramaniam is with the Dept. of ECE, George Washington University, Washington, DC 20052. e-mail:{hongsik,choi,suresh}@seas.gwu.edu.

This work was supported in part by the DARPA under grant N66001-00-18949 (co-funded by NSA) and by the NSF under grants ANI-9973098 and ANI-9973111.

switch to the backup path joining the two end-nodes. In path protection, all affected connections are notified of the link failure, and they switch to the backup paths. Both link and path protection techniques in rings require the reservation of 50% of the total capacity for protection purposes.

More recently, attention has focused on mesh networks partly because of the increased flexibility they provide in routing connections, and partly because the natural evolution of network topologies leads to a mesh-type topology. While protection in mesh networks can potentially be more efficient, it is more complex as well because of the multiplicity of routes which can be used for recovery. Approaches for protecting link failures in mesh netwokks can be found in the recent literature and are briefly reviewed here.

One approach is to use ring-like protection mechanisms by embedding cycles on a given mesh topology. Suppose the network is represented by a directed graph (digraph). Recovery from single-link failures requires the graph to be 2-edge connected,[1] so let us assume that a 2-connected digraph is given. In the *double cycle cover* method of [4], [5], the links of the digraph are covered by two directed cycles such that each link is covered by a cycle in each direction exactly once. A set of cycles that has this property can be found in polynomial time for planar graphs [5] (i.e., graphs that can be drawn on a plane without intersecting edges), but no known polynomial-time algorithm is known for non-planar graphs. On each link, exactly half of the fibers are set aside for protection and half are used for working traffic. Consider the undirected link AB (that includes the directed links AB and BA), and suppose that it is a part of two cycles C_1 and C_2, where C_1 is a cycle that includes directed link AB and C_2 is a cycle that includes directed link BA. Then, all the working fibers from A to B are backed up by the protection fibers from A to B on cycle C_2, and all working fibers from B to A are backed up by the protection fibers from B to A on cycle C_1. Note that this is fiber-based recovery since whole working fibers are backed up by a set of protection fibers. The advantage of this technique lies in the fact that the protection switches can be pre-configured, and no signaling is required upon failure of a link.

Apart from the drawback of not being able to guarantee recovery when the graph is non-planar, the above technique has the disadvantage of requiring wavelength conversion when there is a single fiber in each direction of a link [6]. Accordingly, another method of link protection was presented in [6]. In this method, instead of forming cycles, a 2-connected directed subgraph H that covers all nodes is obtained. Another subgraph H' which is similar to H except that the directions of the edges are reversed is also immediately obtained. On each fiber, half of the wavelengths are working and the other half are reserved for protection. Furthermore, the wavelengths that are reserved for protection in the edges in H are the working wavelengths in H' and vice versa. Then, a failure of an undirected link AB can be recovered as follows. Suppose the working wavelengths on directed link AB belong to H and those on directed link BA belong to H'. The working wavelengths on directed link AB are recovered using the protection wavelengths on directed path AB in subgraph H'. Similarly, the working wavelengths on directed link BA are recovered using protection wavelengths on directed path BA in subgraph H. This method is applicable to non-planar graphs as well.

[1] A graph is said to be k-edge connected if the removal of any k edges does not disconnect the graph. We will drop the term "edge" and simply refer to it as k-connected henceforth.

A variation of the above method has recently been presented [7]. There, the authors propose a new algorithm to find the backup digraph (i.e., $H \cup H'$). A goal of [7] is to find the minimal backup digraph, i.e., a digraph that contains the smallest number of edges while still covering all the nodes. Then, those edges that are not in the backup digraph need not be allocated protection capacity and the capacity thus freed up may be used for traffic that does not require protection. In spite of that, protection for all working traffic can be guaranteed [7]. Note that the problem of finding the minimal backup digraph contains the problem of finding whether a Hamiltonian cycle[2] exists in a digraph, which is a well-known NP-complete problem [8].

The existing methods for link protection are designed for single link failures. As we explain in the next section, double link failures must also be considered in the context of network survivability. In [7], some work that compares the recovery performance of existing protection methods under double-link failures was presented. However, this begs the question of how protection paths *should* be designed in order to tolerate double-link failures. This topic is our focus in the current paper.

The rest of the paper is organized as follows. Some motivation for considering double-link failures and approaches for handling such failures are given in Section II-B. Algorithms for designing protection paths are given in Section III. Numerical results comparing the algorithms with previous algorithms for single-link failures are presented in Section IV, and the paper is concluded in Section V.

II. Double-Link Failure Recovery

A. Motivation

We first motivate the need for considering double-link failures. Single-link failures are common failure scenarios. Normally, recovery from the failure of a link is completed within a few milliseconds to a few seconds depending on the mechanism used for recovery. However, the time it takes to repair the physical link may be a few hours to a few days. It is certainly conceivable that a second link fails in this duration, thus causing two links to be down at one time. There is yet another reason to consider double-link failures. The graph that represents the physical network topology captures only the connectivity between the nodes, but not the physical routing of the links. The physical routing of links is dictated by right of way which is often obtained from railroad, thruways, and pipeline companies [9]. As an example, a link from New York to Boston and from New York to Washington may be physically routed together for some distance, e.g., along the Lincoln Tunnel. Figure 1 shows a possible physical routing topology and the corresponding graph that represents node connectivities. In this example, a single backhoe accident may lead to the failures of

[2] A Hamiltonian cycle in a graph is a cycle that covers all the graph nodes.

both links AD and BD in the graph topology, which is what protection planning algorithms use to find backup paths.

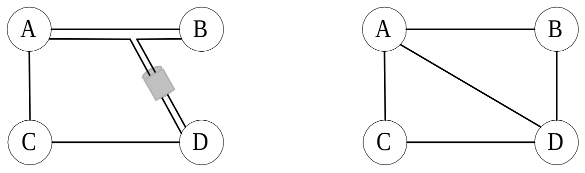

Fig. 1. (a) A physical routing topology, and (b) the graph.

We now discuss some approaches for recovering from double link failures. Let us assume that the second link fails after the recovery from the first failure is complete. The approaches also work when two links fail simultaneously.

B. Double-Link Failure Recovery Approaches

We do not consider path rerouting approaches here, nor dynamic restoration approaches that do not reserve capacity. We focus only on schemes that reroute around the failed links using pre-assigned capacity. However, these schemes require signaling to configure the switches when links fail. We also assume that all working wavelengths on a failed link are rerouted along the same backup path.

Note that for the graph to remain connected when any two edges fail, the graph must be 3-connected. For simplicity of explanation, let us assume this to be the case. When this is not the case, the failure of some edge pairs cannot be tolerated. By Menger's theorem [10], a graph is k-connected if and only if there exist k edge-disjoint paths between every pair of nodes in the graph. It is also possible to find these edge-disjoint paths easily (e.g., by a simple application of the Ford-Fulkerson max-flow algorithm [10]).

The following approaches are possible. The first two require identification of failed links while the third one does not.

1) Method I: Backup paths with link identification - I: Here, two edge-disjoint backup paths, a primary[3] backup path $p_1(e)$ and a secondary backup path $p_2(e)$ are computed for each edge e. The existence of these paths is guaranteed by Menger's theorem, and as mentioned above, the paths can also be easily determined.

When e fails, the primary backup path $p_1(e)$ is used for rerouting. At the same time, all nodes in the network are informed of the failure through signaling. Note that the backup path rerouting is done in parallel with the signaling, and recovery from e's failure may be accomplished before e's failure is broadcast to all nodes. Now, suppose a second edge f fails. This failure is notified to all nodes as before. If the primary backup path $p_1(f)$ does not use e, then $p_1(f)$ is used to reroute the traffic on f, else $p_2(f)$ is used. This is possible because the end-nodes of f know which link has failed previously.

There are two cases possible when f fails: f does not lie on $p_1(e)$, or f lies on $p_1(e)$. In the former case, $p_1(e)$ will continue to be used for e. Note that if $p_1(e)$ and $p_1(f)$ share links, then the protection capacity that must be reserved on the common links is twice the link working capacity because it has to carry the working traffic of both e and f. When f lies on $p_1(e)$ and the information about f's failure reaches the end-nodes of e, these nodes switch the traffic originally on e from $p_1(e)$ to $p_2(e)$ (which is disjoint with f). Observe that the knowledge of which links lie on a backup path is necessary to carry out this process.

The amount of signaling that this scheme requires is roughly equal to what is necessary in path rerouting, but possibly, fewer switches may need to be configured in the link rerouting scheme. Moreover, backup capacity must be pre-allocated on both $p_1(e)$ and $p_2(e)$ since each of them may be active at different times, and the capacity cannot be shared between them because they are link-disjoint.

2) Method II: Backup paths with link identification - II: This is similar to the previous scheme except for the following. Suppose, without loss of generality, that $p_1(f)$ does not use e. When f fails, and if f lies on $p_1(e)$, $p_1(f)$ is used to back up *both* the working traffic on f as well as the backup traffic rerouted on $p_1(e)$. Thus, the working traffic originally routed on e is now routed on $p_1(e) - \{f\} \cup p_1(f)$. This scenario is shown in Figure 2.

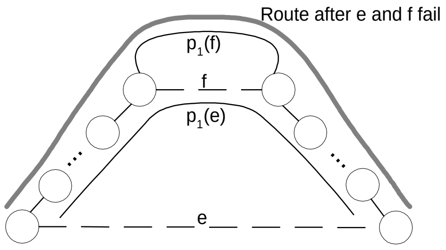

Fig. 2. Rerouting with link identifi cation - II.

Note that this scheme still requires that the end-nodes of f know that e has failed, though the failure of f need not be known by nodes other than its end-nodes for recovery to be completed. The cost for this is the increased path length for the backup of the traffic originally routed on e. Here, the backup capacity that is required on $p_1(f)$ is twice the working capacity. This capacity is required even if the links on $p_1(f)$ are not used as backup for any other link (unlike in the previous case). This is because these links carry e's as well as f's working traffic.

3) Method III: Backup paths without link identification: In this approach, a single backup path $p(e)$ is pre-computed for each link e. Suppose for the moment that the backup path $p(f)$ for every link $f \in p(e)$ does not contain e. Suppose e fails first. Then, the traffic on e is rerouted along $p(e)$. Now, if f fails, the working traffic on f and any rerouted traffic on f (in this case, the rerouted traffic from e) are both switched to $p(f)$ from f. Since $p(f)$ does not use e, this rerouting would be successful.

Observe that no signaling is necessary to inform the network nodes of a link's failure; the failure of a link need only be detected at the end-nodes of that link. In this scheme, the protection capacity that must be allocated on a link, say f, is computed using the following rules:

- If f is not part of any backup path, then no protection capacity is needed.
- If f is part of the backup path of exactly one link, say e,

[3]This is not to be confused with a primary path which is sometimes used to refer to a working path.

and if e is not on any link's backup path, then the protection capacity needed on f is equal to the working capacity on any link. On the other hand, if e lies on the backup path(s) of one or more links, then twice the link working capacity must be reserved on f. This is because, if e is carrying rerouted traffic from some other link failure when it fails, then that rerouted traffic *and* the working traffic on e must *both* be rerouted along $p(e)$ which contains f.

- If f is on the backup paths of two or more links, then twice the link working capacity must be reserved for protection on f.

It is possible to obtain such rules for the previous two schemes as well, since all paths are pre-computed, and the algorithm for re-routing is also pre-decided.

The advantage of this approach is that no failed link identification is necessary (except by the end-nodes of the failed link). As in the previous scheme, the rerouted traffic when two links fail may have to traverse many links.

An important consideration in this scheme is how the backup paths must be computed. Recall that in the previous two schemes, one only needed to compute two link-disjoint backup paths for each link, and this is easily done.[4] In the third approach, however, we need only a single backup path for each link, but the backup paths of the various links must satisfy the special property that was assumed earlier; namely, if $p(e)$ contains f, then $p(f)$ must not use e.

It is not clear if such a set of backup paths can be computed even if a graph is 3-connected. Our goal in the next section is to address the problem of computing such backup paths.

III. An Algorithm for Backup Paths

In this section, we formulate the problem of computing backup paths as required by Method III described in Section II-B.3, and present an algorithm to compute the backup paths. We assume that a graph G representing the network is given, and assume that any two arbitrary links in G may fail.

A. Problem Formulation

As mentioned earlier, there exist double-link failures that cannot be tolerated by any algorithm if the graph is not 3-connected. However, it is not known if backup paths can be pre-computed for every link as required by Method III, even if the graph is 3-connected. In any case, many practical network topologies are not 3-connected (including the ones we consider in the next section). Therefore, we consider the more practical problem of computing backup paths such that the maximum possible number of double-link failures can be tolerated.

Formally, we seek an answer to the following problem:
Maximum Arbitrary Double-Link Protection Problem (MADPP): Given G, find a backup path $p(e)$ for each link e such that the set $F = \{\{e, f\} | e \in p(f) \text{ and } f \in p(e)\}$, has minimum cardinality, where e and f are arbitrary links in G.

[4]3-connectivity is assumed here. If not, then no scheme can tolerate the failure of all double-link failures, but it is possible to find two edge-disjoint paths for those links that do have two edge-disjoint paths in polynomial time by using the max-flow algorithm.

Note that F is exactly the set of link pairs whose failures cannot be tolerated by Method III. We conjecture that this problem is NP-hard. In the following, we present a heuristic algorithm called as the *Maximum Arbitrary Double-Link Protection Algorithm (MADPA)*. This algorithm will be shown to give excellent results in Section IV.

B. The Maximum Arbitrary Double-Link Protection Algorithm (MADPA)

Ours is a recursive algorithm whose operation can be summarized as follows. It works by contracting the graph G according to a set of rules, computing backup paths for the links in the contracted graph, and then mapping these backup paths back to the original graph.

Since almost all existing network topologies are 2-connected, we will assume that G is 2-connected throughout this section. The algorithm is organized in three phases.

1) Phase 1: Pre-processing: In this phase, G is pre-processed in the following way. If a node v has only two adjacent nodes u and w, then delete node v and merge the edges $(u, v), (v, w)$ to form a single edge (u, w). If G has such a node, note that the failure of both edges (u, v) and (v, w) cannot be tolerated by any algorithm. Let the graph after this pre-processing be G_0 and observe that 2-connectivity is still maintained after this phase.

2) Phase 2: Contraction: In this phase, we contract the graph G_0 in a succession of steps using a set of prioritized rules. The rules are the result of an extensive investigation of various cases that arise in the computation of backup paths. At each step of this phase, we have a graph G_i, and a new contracted graph G_{i+1} is obtained by using one of the 4 rules given below. The rules are prioritized such that Rule 1 is applied first if possible, else Rule 2 is applied if possible, and so on. The contracting phase stops when, for some k, G_k has only two nodes. The specific sequence of rules that was used in contracting G_0 to G_k is also kept track of in this phase.
Rule 1: If there is a pair of nodes u and v connected by two or more edges in G_i, we form G_{i+1} by merging u and v into a single node and removing the edge (u, v). This is illustrated in Figure 3.

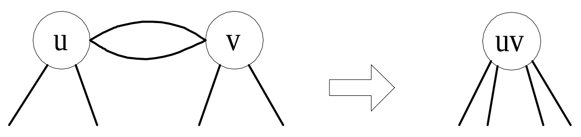

Fig. 3. Contraction by Rule 1.

Rule 2: If there are three nodes $u, v,$ and w such that the degree of each node is *exactly* three and $u, v,$ and w are neighbors of each other, then G_{i+1} is obtained as follows. $u, v,$ and w are merged into a single node called uvw with three edges incident to it as shown in Figure 4.
Rule 3: If there are three nodes $u, v,$ and w such that they are adjacent to each other, and the degree of (at least) one of the three nodes is greater than 3, the contraction is done through a two-step process. Suppose, without loss of generality, that w

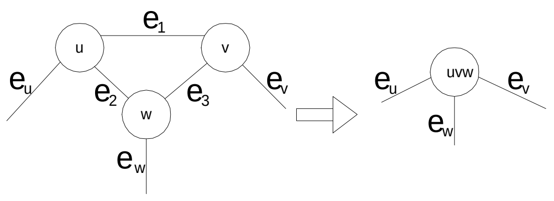

Fig. 4. Contraction by Rule 2.

has degree greater than 3. In the first step, G_i is contracted by merging u and v into a single node uv, resulting in a new graph which we call $G_{i+0.5}$. In the second step, Rule 1 is applied to the pair of edges between the nodes uv and w in $G_{i+0.5}$ to form G_{i+1}. This rule is illustrated in Figure 5 where the degree of w is assumed to be 4.

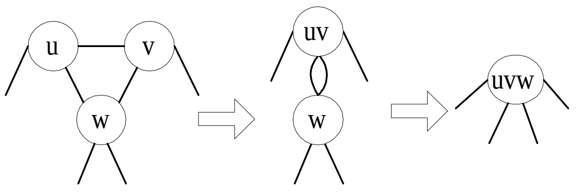

Fig. 5. Contraction by Rule 3.

Rule 4: Find an arbitrary edge (u, v) and obtain G_{i+1} by merging u and v into a single node uv, and deleting the edge (u, v). Other edges that are incident to either u or v in G_i are incident to node uv in G_{i+1}.

As noted earlier, the contracting phase stops when there are exactly two nodes in G_k for some k.[5] Before describing the third phase, which is the *expansion phase*, we make a couple of observations about G_k.

Lemma 1: G_k is 2-connected.

Proof: We show this by induction on i. Suppose G_i is 2-connected for some $0 \leq i < k$, i.e., the deletion of any edge does not disconnect the graph. Then, it is easy to see that the application of any of the 4 contracting rules preserves 2-connectivity, and therefore G_{i+1} is 2-connected. In observing this, keep in mind that if two nodes remain, the contracting phase stops. ■

Lemma 2: G_k has two nodes with two or more edges between them.

Proof: This follows from the facts that the contracting phase stops when G_k has two nodes and that G_k is 2-connected (as shown in Lemma 1). ■

In the expansion phase of the algorithm, described next, we present how assigned backup paths are mapped from G_{i+1} to G_i.

3) Phase 3: Expansion: In this phase, backup paths are first assigned to G_k, and G_k is expanded in a sequence of steps to G_0 by reversing the sequence of rules used in the contraction phase to obtain G_k. In the following, we assume that a backup path assignment for some G_{i+1}, $0 < i + 1 \leq k$ is given, and we show how G_{i+1} is expanded to G_i and backup paths are mapped from G_{i+1} to G_i. We also show how backup paths for the edges that appear in G_i but did not exist in G_{i+1} are assigned. Recall that our goal is to assign backup paths as required by Method III, i.e., if the backup path of link e uses f, then we would like the backup path of f not to use e. The backup path assignment during the expansion phase is done with this goal in mind.

If G_{i+1} was obtained from G_i by using contracting rule j ($j = 1, 2, 3, 4$), then the expansion is done by using the expansion rule j described below.

In the following, we let $p_i(e)$ denote the backup path for edge e in G_i. In all of the rules below, if $p_{i+1}(e)$ does not pass through any nodes that are merged in getting G_{i+1} from G_i, then $p_i(e)$ is kept the same as $p_{i+1}(e)$.

Rule 1: In this case, G_{i+1} is expanded to G_i by making two nodes u, v and inserting the edges between u and v that were deleted in the contraction phase. Let the number of edges between u and v be $m \geq 2$ (see contraction Rule 1).

This expansion rule is illustrated in Figure 6. We distinguish between the cases $m = 2$ and $m > 2$.

Case 1: $m = 2$.

If $p_{i+1}(e)$ passes through node uv, then $p_i(e)$ is set to $p_{i+1}(e) \cup e_1$ (if e_1 is necessary; note that some backup paths may not need to use e_1 as shown in Figure 6), otherwise set $p_i(e)$ to $p_{i+1}(e)$. Then, we set $p_i(e_1) = \{e_2\}$. For $p_i(e_2)$, any path between nodes u and v (other than e_1) is used as a backup path. Such a path must exist because G_{i+1} is 2-connected.

Case 2: $m > 2$.

Let $e_1, e_2, \ldots, e_m$ be the m edges between nodes u and v. If $p_{i+1}(e)$ passes through node uv, then $p_i(e)$ is set to $p_{i+1}(e) \cup \{e_1\}$ (if e_1 is necessary), otherwise set $p_i(e)$ to $p_{i+1}(e)$. For the edges $e_1, e_2, \ldots e_m$, the backup paths are assigned as follows: $p_i(e_1) = \{e_2\}, p_i(e_2) = \{e_3\}, \ldots p_i(e_{m-1}) = \{e_m\}, p_i(e_m) = \{e_1\}$.

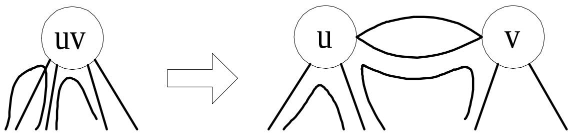

Fig. 6. Expansion by Rule 1. Case 1 is shown.

Our backup path assignment according to Rule 1 has this property: if backup paths are assigned successfully to tolerate the maximum number of double-link failures in G_{i+1}, then the backup paths in G_i also tolerate the maximum number of double-link failures. This can be easily observed from the way the backup path assignment was done using Rule 1 – if there are two edges e and f between u and v in G_i, and if $p(e)$ uses f, then we made sure that $p(f)$ does not use e.

Rule 2: In G_i, let us denote edge (u, v) by e_1, edge (v, w) by e_2 and edge (u, w) by e_3. Let the other three edges incident at nodes $u, v,$ and w be denoted by $e_u, e_v,$ and e_w, respectively. Note that these edges were the only ones incident to node uvw in G_{i+1}.

We observe that the backup path for edge e_u in G_{i+1} uses exactly one of the two edges e_v and e_w. A similar observation

[5] It is easy to see that we will end up with two nodes eventually if the contracting rules are applied to any 2-connected graph G_0.

can be made for e_v and e_w as well. Without loss of generality, let us assume that $p(e_u)$ uses e_v, $p(e_v)$ uses e_w, and $p(e_w)$ uses e_u.

Among the edges in G_{i+1}, some backup paths do not pass through node uvw, and for these edges, the backup paths are the same in G_i as in G_{i+1}. Every edge in G_{i+1} whose backup path passes through node uvw uses exactly two of the edges $e_u, e_v,$ and e_w. Of these edges, let e_a denote an edge such that $p(e_a)$ that uses e_u and e_v, e_b denote an edge such that $p(e_b)$ uses e_v and e_w, and e_c denote an edge such that $p(e_c)$ uses e_u and e_w. The notation for the labeling of edges is illustrated in Figure 7.

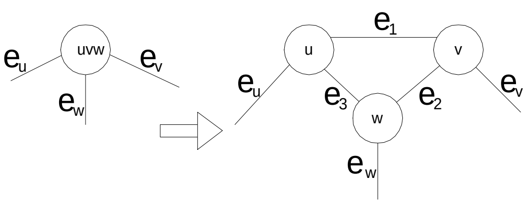

Fig. 7. Notation for expansion by Rule 2.

We now provide the backup path assignment for the edges in G_i:

$p_i(e_u) = \{e_2, e_3\} \cup p_{i+1}(e_u)$,
$p_i(e_v) = \{e_2\} \cup p_{i+1}(e_v)$,
$p_i(e_w) = \{e_2, e_1\} \cup p_{i+1}(e_w)$,
$p_i(e_1) = \{e_u\} \cup p_{i+1}(e_u)$,
$p_i(e_2) = \{e_1, e_3\}$,
$p_i(e_3) = \{e_1, e_v\} \cup p_{i+1}(e_v)$,
$p_i(e_a) = \{e_1\} \cup p_{i+1}(e_a)$,
$p_i(e_b) = \{e_2\} \cup p_{i+1}(e_b)$, and
$p_i(e_c) = \{e_3\} \cup p_{i+1}(e_c)$.

Once again, if backup paths in G_{i+1} have been optimally designed, then the backup paths in G_i are also optimal in the sense that G_i is vulnerable to no more double-link failures than G_{i+1}. This is because, as in expansion by Rule 1, we made sure that if $p(e)$ uses f, then $p(f)$ does not use e for any two edges e, f. It is obvious that this condition is satisfied for the backup paths of $e_1, e_2, e_3, e_u, e_v,$ and e_w. For e_a, the backup path $p_i(e_a)$ uses e_1; let us now show that $p_i(e_1)$ does not use e_a assuming backup paths were computed successfully in G_{i+1}. In G_{i+1}, $p_{i+1}(e_a)$ uses e_u (by definition of e_a), so $p_{i+1}(e_u)$ does not use e_a. From the assignment of $p_i(e_1)$ given above, it follows that $p_i(e_1)$ does not use e_a. Similar properties can be shown for e_b and e_c as well.

Rule 3: Recall that the contraction Rule 3 was a two-step process wherein contraction Rule 1 followed the merging of two nodes u and v (see Figure 5). Without loss of generality, suppose that G_{i+1} obtained from G_i using contraction Rule 3 is as shown in Figure 8. The backup paths in $G_{i+0.5}$ are assigned by using expansion Rule 1, Case 1. Suppose that some backup paths in $G_{i+0.5}$ use e_2 (and none use e_3) after expansion Rule 1, Case 1 is applied to G_{i+1}. Then, $p_{i+0.5}(e_2) = \{e_3\}$.

We now explain how backup paths are assigned in G_i. The reader is referred to Figure 9. For all edges e in G_i except $e_1, e_2,$ and e_3, we set $p_i(e) = p_{i+0.5}(e) \cup \{e_1\}$, if e_1 is necessary, else set $p_i(e) = p_{i+0.5}(e)$. Now, set $p_i(e_2) = \{e_3, e_1\}$,

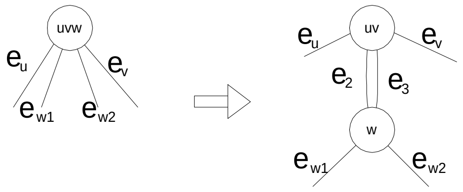

Fig. 8. Expansion by Rule 3, Step 1.

and $p_i(e_3) = p_{i+0.5}(e_3)$. For $p_i(e_1)$, we try to find a path q between nodes u and w in G_i such that it does not use e_1. If such a path is available, set $p_i(e_1) = q \cup \{e_3\}$. Otherwise, we simply set $p_i(e_1) = \{e_2, e_3\}$ and note that the failure of e_1 and e_2 together cannot be tolerated by our algorithm. This is because $p_i(e_2)$ uses $p_i(e_1)$ and vice versa.

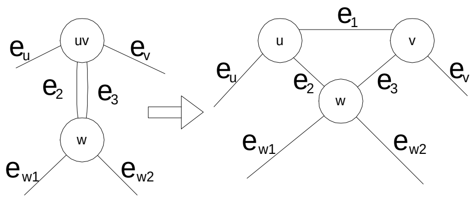

Fig. 9. Expansion by Rule 3, Step 2.

Rule 4: If contraction of G_i was done using Rule 4, then we assign backup paths for G_i as follows. Recall that contraction Rule 4 merged two adjacent nodes u and v. Let $e = (u, v)$ be the edge between u and v in G_i. For the edges in G_{i+1}, the backup paths in G_i simply add e to their backup paths in G_{i+1} if necessary, otherwise use their backup paths in G_{i+1} as they are. Let Q be the set of edges that added e to their backup paths. We then try to find a backup path for e without using any edge in Q. If such a path is found, then our backup path assignment is successful. If not, find some backup path for e that uses at least one edge from Q. If this case happens, then the failure of e and any of the edges from Q that are used in $p_i(e)$ cannot be tolerated.

Once the backup paths for G_0 are assigned, they are mapped to backup paths in G in a straightforward way. Suppose node v was deleted and edges (u, v) and (v, w) were merged into edge (u, w) in the pre-processing phase. Then, if a backup path uses edge (u, w) in G_0, they are mapped to the same path in G except that (u, v) and (v, w) are used in G. For the backup paths of (u, v) and (v, w), the backup path of (u, w) in G_0 is used for one of them, say (u, v). For (v, w), we simply find an arbitrary backup path. Note that any such path uses (u, v) so the failure of (u, v) and (v, w) would not be tolerable.

This completes the description of our algorithm for computing backup paths for Method III. We present some numerical results in the next section that demonstrates the effectiveness of our algorithm.

IV. Numerical Results

In this section, we present some numerical results comparing the performance of the various algorithms. For Method I and

Method II described in Section II-B, we computed two edge-disjoint backup paths (excluding e) between the end-nodes of every edge e if such paths are possible, otherwise we computed a single backup path. A single path was always possible for all of the networks that we considered because all of them are 2-connected.

For Method III, we implemented our algorithm (MADPA), the backup paths resulting from the WDM loopback recovery method of [6] (WL), and a shortest backup path algorithm (SP). In the shortest backup path algorithm, a shortest path to be used as backup is found between the end-nodes of every edge. We do not present results for the backup paths resulting from the double-cycle cover method [5] because that algorithm is guaranteed to provide a backup path for every edge only when the graph is planar. As it turns out, two of the networks that we present results for are not planar.

The following performance measures are used in the comparison:

- Protection capacity: This is the capacity that must be reserved for protection on all the links. Some of the recovery methods are not capable of performing individual wavelength recovery, but perform fiber recovery. These methods require that an entire fiber in each direction be dedicated for working traffic, if wavelength conversion has to be avoided [6]. We measure capacity by the number of network links that do not require backup capacity, the number of links that require 100% backup capacity (i.e., a backup capacity equal to 100% of link working capacity), and the number of links that require 200% backup capacity. The backup capacities are obtained according to the procedure outlined in Section II-B.
- Restorability: The restorability is the fraction of the number of double-link failures that can be tolerated. We consider the order of failure of the two links here, hence the number of failures is $L(L-1)$ for arbitrary double-link failures, where L is the number of links. Note that if the graph is not 3-connected, some double-link failures cannot be tolerated by *any* algorithm. This number can be easily computed by finding the number of edge cut-sets of size 2, i.e., finding the number of pairs of edges whose deletion would disconnect the graph.
- Worst-case and average hop-length: Once again, the order of failure of the links is taken into account. The worst-case hop-length is the largest number of links used in rerouting, over the $L(L-1)$ possible failures, i.e., the number of extra links that are used to reroute the traffic when two links fail. The average hop-length is the average over all these failures.

We note that Methods I and II may use less capacity and use shorter hop-lengths, but require considerably more signaling. At this point of time, we are unable to present results on recovery time because of the lack of an accurate model for computing recovery time. We hope to present these in the future.

We report results for three example network topologies that have been studied in the literature. These network topologies are shown in Figures 10, 11, and 12.

The following notation is used in the tables. The average hop-length is denoted by $\bar{H}$, the worst-case hop-length by

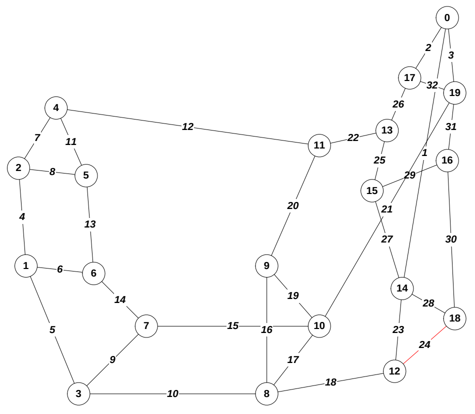
Fig. 10. The ARPANET network.

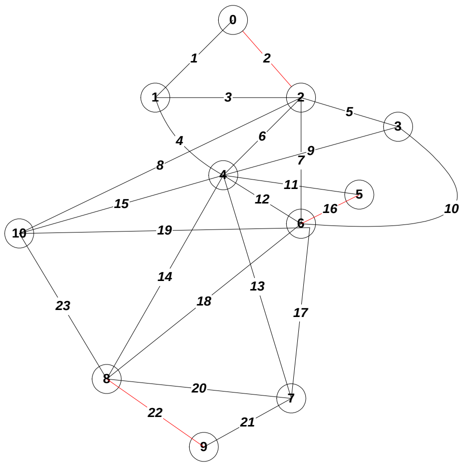
Fig. 11. The NJ LATA network.

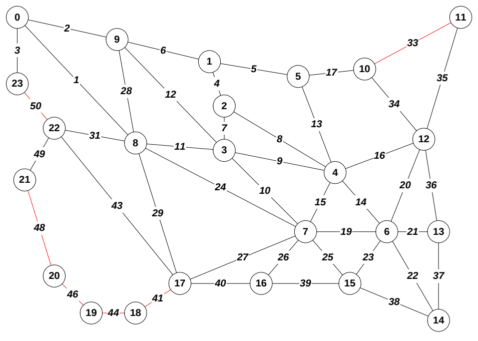
Fig. 12. The National network.

$H_{\max}$, and the number of links with $0, 100,$ and 200% backup capacities by $B_0, B_{100},$ and B_{200}. The number of restorable double-link failures by the given algorithm is denoted by R. For the algorithms, I and II are used to denote Methods I and II, respectively, and WL, SP, and MADPA denote the use of these algorithms for computing link backup paths in Method III.

The results for the 32-link ARPANET network is given below in Table I. The number of link pairs (taking order into account) is 992, and the number of ordered two-edge cut-sets is 0, leaving all 992 ordered double-link failures potentially recoverable.

TABLE I
COMPARISON OF ALGORITHMS: ARPANET.

Metric	I	II	WL	SP	MADPA
H	6.8	6.4	14.0	6.4	12.2
$H_{\max}$	16	19	23	11	18
R	992	992	938	938	980
B_0	0	0	2	2	0
B_{100}	0	0	2	4	0
B_{200}	32	32	28	26	32

We next present results for the NJ LATA network in Table II. This network has 23 links, and the number of ordered link pairs is 506. The number of ordered 2-edge cut-sets is 6, leaving 500 ordered pairs of link failures to be potentially recoverable.

TABLE II
COMPARISON OF ALGORITHMS: NJ LATA.

Metric	I	II	WL	SP	MADPA
H	3.9	3.9	5.3	4.1	7.6
$H_{\max}$	7	6	7	6	12
R	500	500	482	484	500
B_0	0	0	5	2	0
B_{100}	10	2	6	11	1
B_{200}	13	21	12	10	22

Finally, Table III shows the results for the National network. This network has 44 links and therefore 1892 ordered link pairs. The number of ordered 2-edge cuts is 24 leaving a total of 1868 ordered double-link failures to be potentially recoverable.

TABLE III
COMPARISON OF ALGORITHMS: NATIONAL NETWORK.

Metric	I	II	WL	SP	MADPA
H	5.0	5.0	11.0	5.0	8.3
$H_{\max}$	10	9	19	8	17
R	1868	1868	1818	1828	1868
B_0	0	0	5	3	0
B_{100}	8	5	0	8	4
B_{200}	36	39	39	33	40

We make the following observations from the numerical results. Methods I and II provide 100% protection against double-link failures, and have among the lowest average and maximum hop lengths. Interestingly, Method II results in a smaller $H_{\max}$ for two of the networks, whereas one might expect Method I to have shorter maximum hop-lengths. However, both of these methods may require extensive signaling and therefore much longer restoration times when compared with Method III. In Method III, as expected, SP results in shorter backup paths than WL and MADPA. MADPA provides better protection against double-link failures than WL and SP. In fact, the effectiveness of MADPA can be gauged from the fact 100% restorability was possible in two of the topologies (98.8% in the ARPANET). However, it also requires the most backup capacity of the three backup path computation algorithms.[6] WL required the lowest backup capacity but also gave the lowest restorability in all the topologies and the largest hop-lengths in two of the topologies. We note, however, that WL and SP were not designed for double-link failure protection.

V. CONCLUSIONS AND FUTURE WORK

Network survivability is a crucial requirement in high-speed optical networks. Typical approaches of providing survivability have considered the failure of a single component such as a link or a node. In this paper, we motivated the need for considering double-link failures and presented some approaches for handling such failures.

A heuristic algorithm that pre-computes backup paths for links in order to tolerate double-link failures was then presented. Numerical results comparing the performance of our algorithm with other approaches suggests that it is possible to achieve almost 100% recovery from double-link failures with a modest increase in backup capacity.

In this work, we have assumed that any two arbitrary links may fail in any order. Future work may consider other failure models. Pre-computing backup paths that minimize the capacity requirement under double-link failures is another interesting direction of study. Exploring the trade-off between restorability and backup capacity is yet another possible topic for future study.

REFERENCES

[1] Ornan Gerstel and Rajiv Ramaswami. Optical layer survivability: A services perspective. *IEEE Communications Magazine*, 38(3):104–113, March 2000.
[2] R. Ramamurthy, Z. Bogdanowicz, S. Samieian, D. Saha, B. Rajagopalan, S. Sengupta, S. Chauduri, and K. Bala. Capacity performance of dynamic provisioning in optical networks. *IEEE/OSA J. Lightwave Tech.*, 19(1):40–48, Jan. 2001.
[3] R. Ramaswami and K. N. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufmann, 1998.
[4] G. Ellinas and T. Stern. Automatic protection switching for link failures in optical networks with bidirectional links. In *Proc. GLOBECOM*, pages 152–6, Nov. 1996.
[5] G. Ellinas, G. Halemariam, and T. Stern. Protection cycles in mesh WDM networks. *IEEE J. Sel. Areas Comm.*, 18(10):1924–37, Oct. 2000.
[6] M. Medard, S. G. Finn, and R. A. Barry. WDM loop-back recovery in mesh networks. In *Proc. INFOCOM*, pages 752–759, March 1999.
[7] S. S. Lumetta, M Medard, and Y.-C. Tseng. Capacity versus robustness: a tradeoff for link restoration in mesh networks. *IEEE/OSA J. Lightwave Tech.*, 18(12):1765–75, Dec. 2000.

[6] Note that the backup capacities for SP and WL were obtained considering double-link failures, and are therefore more than corresponding results for single-link failures.

[8] M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Disc. Math.*, 1(2):216–227, June 1980.

[9] J. Strand and A. L. Chiu. Issues for routing in the optical layer. *IEEE Communications Magazine*, 39(2):81–87, Feb. 2001.

[10] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. American Elsevier Publishing, 1976.