# Lightpaths Routing for Single Link Failure Survivability in IP-over-WDM Networks

Muhammad Javed, Krishnaiyan Thulasiraman, and Guoliang (Larry) Xue

*Abstract:* **High speed all optical network is a viable option to satisfy the exponential growth of internet usage in the recent years. Optical networks offer very high bit rates and, by employing technologies like internet protocol over wavelength division multiplexing (IP-over-WDM), these high bit rates can be effectively utilized. However, failure of a network component, carrying such high speed data traffic can result in enormous loss of data in a few seconds and persistence of a failure can severely degrade the performance of the entire network. Designing IP-over-WDM networks, which can withstand failures, has been subject of considerable interest in the research community recently. Most of the research is focused on the failure of optical links in the network. This paper addresses the problem of designing IP-over-WDM networks that do not suffer service degradation in case of a single link failure. The paper proposes an approach based on the framework provided by a recent paper by M. Kurant and P. Thiran. The proposed approach can be used to design large survivable IP-over-WDM networks.**

*Index Terms:* **Internet protocol over wavelength division multiplexing (IP-over-WDM), lightpath routing, optical networks, survivability.**

## I. INTRODUCTION

Developments in the wavelength division multiplexing (WDM) technology have made it possible to effectively utilize the huge bandwidth offered by optical networks. One such development is the ability to implement internet protocol (IP) directly over the optical network using IP routers and optical crossconnects (OXC). This combination allows a more flexible use of optical networks, which were initially limited to point-to-point communication. Since most of the current end user communications are based on the transmission control protocol (TCP)/IP, the option of implementing IP directly over the optical network has attracted significant interest from the research community and such networks are commonly referred to as IP-over-WDM networks.

A commonly proposed method of implementing IP-over-WDM is to embed an IP topology, referred to as *logical topology*, in a physical WDM topology, commonly called *physical topology*. The embedding involves finding a *lightpath* for an IP (logical) link in the physical topology, which connects the two end points of the logical link [1]. A *lightpath* is an all optical path established by the allocation of a wavelength between a source and a destination. A lightpath, once established, does not require processing or buffering at intermediate nodes and quite possibly no intermediate optical-electrical conversions [1]. Usually a single fiber carries multiple lightpaths and all of them get disconnected if the fiber carrying them fails. Even a single fiber failure can result in enormous loss of data, usually in the order of gigabits per second [2]. Therefore, the entire network may suffer severe degradation of service if the failure persists for a longer duration.

Unfortunately, fiber failures are very common and are mainly due to fiber cuts [3]. Frequent failures in optical networks demand mechanisms that would allow the network to tolerate such failures. A network capable of surviving single or multiple failures is usually referred to as "*survivable*". There are two widely discussed methodologies for providing such capability; protection and restoration. Protection requires pre-computed alternate paths for the failed components and restoration requires computation of alternate paths after the failure has occurred. Protection provides very fast recovery time but spare capacity must be reserved when network is being designed, which makes the network less efficient in terms of capacity utilization. Restoration is generally slow but does not require dedicated capacity at the design time. Upon failure of a network component (link, router, or OXC), a restoration scheme finds alternate paths in the spare capacity of the network. The choice between using protection or restoration, to make the network survivable, largely depends upon the desired recovery speed, cost involved in terms of spare capacity and equipment dedication, and the network layer being considered [4]. Some networks may even implement both protection and restoration strategies but at different network layers [4].

In IP-over-WDM networks survivability can be provided at the WDM layer (protection) or at the IP layer (restoration) [2]. At WDM layer, survivability is provided by establishing backup lightpaths for the primary (or working) lightpaths, which do not use any of the fibers that are on the paths used by the primary lightpaths. The traffic on the primary lightpath is always routed on its corresponding backup lightpath. This may require investment in installation of additional fibers to provide diverse routes. To make the network more capacity efficient, a single backup lightpath may be shared by several primary lightpaths (which are not expected to fail simultaneously). However, providing diverse routes may be difficult and expensive due to geographical and right of way constraints. Providing survivability at the IP layer does not require dedicated backup lightpaths. Instead the network is provisioned with some additional capacity, and backup paths are found for the failed lightpaths within this additional capacity at the time of failure. Both protection and restoration mechanisms have the capability to make a network survivable but it is not clear which would be a better choice [2]. This paper focuses only on the protection strategies implemented at the IP layer.

Manuscript received May 16, 2007.

M. Javed and K. Thulasiraman are with the Department of Computer Science at University of Oklahoma, Norman, OK, email: { javed, thulasi}@ou.edu.

G. L. Xue is with School of Computing and Informatics at Arizona State University (ASU), email: xue@asu.edu.

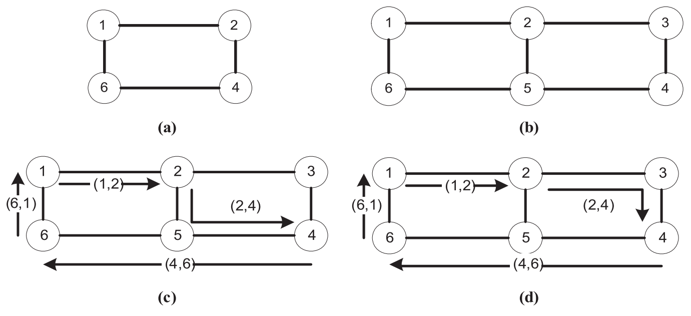


Fig. 1. Illustration of mapping and survivability: (a) Logical topology, (b) physical topology, (c) an unsurvivable mapping, and (d) a survivable mapping.

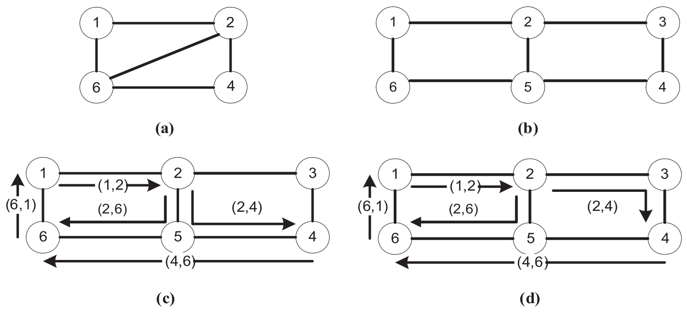


Fig. 2. Illustration of mapping and survivability: (a) Logical topology, (b) physical topology, (c) an unsurvivable mapping, and (d) a survivable mapping.

In IP-over-WDM networks, IP (logical) links are embedded in the physical WDM topology as lightpaths. When a network component fails, all the lightpaths passing through this component are disrupted. The IP routers adjacent to the failed component can detect the failure and initiate the protocol to find alternate routes for the failed lightpaths. These routers will find alternate routes, if the logical topology stays connected after the failure. If the failed network component is a link and the logical topology remains connected after the failure, such an embedding of logical links in the physical topology is called *link-survivable mapping*. Similarly, if the failed component is an IP router or an OXC (or simply node failure) and the logical topology remains connected, such a embedding is called *node-survivable mapping*. A network can be protected against node failures by providing redundant equipment. Since links failures are much more common than node failures, this paper focuses on making logical topology survivable, in case of a *single* physical link failure.

A necessary pre-condition for a network to be able to survive a link failure is that both physical and logical networks must be at least two-connected. If this condition is fulfilled, then the logical topology can be made survivable by requiring that some or all of the logical links are mapped in the physical topology in a disjoint manner (i.e. paths followed by the mapped links do not have a physical link in common). This requirement ensures that a single physical link failure will not disconnect the logical topology. Since the logical topology is two-connected, it is always possible for the IP routers to find an alternate path between the affected nodes as illustrated in the following example.

The logical topology consists of a ring 1-2-4-6 as shown in Fig. 1(a). The physical topology is shown in Fig. 1(b). Fig. 1(c) shows a possible mapping of the logical topology. Here the logical links (1, 2), (2, 4), (4, 6), and (6, 1) are mapped to the physical paths 1-2, 2-5-4, 4-5-6, and 6-1, respectively. However, this mapping is not survivable because if the physical link 4-5 fails, the logical links 2-4 and 4-6 fail, isolating node 4. Fig. 1(d) shows another possible mapping for the same logical and physical topologies. If the logical links (1, 2), (2, 4), (4, 6), and (6, 1) are mapped respectively to the physical paths 1-2, 2-3-4, 4-5-6, and 6-1, the routing is survivable. Any physical link failure does not disconnect the logical topology because these paths are link disjoint. For example, if the physical link 2-3 fails, the logical link 2-4 fails but the topology remains connected.

Fig. 2(a), a more general logical topology is considered. The physical topology is shown in Fig. 2(b). Fig. 2(c) shows a possible mapping for the logical topology in the physical topology. Here, the logical links (1, 2), (2, 4), (4, 6), (6, 1), and (2, 6) are mapped to the physical paths 1-2, 2-5-4, 4-5-6, 6-1, and 2-5-6, respectively. However, this mapping is not survivable because the physical link 4-5 is being used by logical links (2, 4) and (4, 6) and if the physical link 4-5 fails, the logical links 2-4 and 4-6 fail, isolating node 4. Fig. 2(c) also shows that the physical link 5-6 is common to the mapping of logical links (4, 6) and (2, 6). But it can be observed that when the physical link 5-6 fails, logical links (4, 6) and (2, 6) fail but the logical topology still remains connected. Fig. 2(d) shows another possible mapping of the logical topology. Here logical links (1, 2), (2, 4), (4, 6), (6, 1), and (2, 6) are mapped respectively to the physical paths 1-2, 2-3-4, 4-5-6, 6-1, and 2-5-6 and it can be observed that, even though all the paths are not link disjoint, any single physical link failure does not disconnect the logical topology. For example, if the physical link 5-6 fails, the logical links 4-6 and 2-6 fail but the topology remains connected.

The above examples illustrate some of the difficulties involved in finding survivable mappings of logical topologies. If the logical topology is a cycle, the problem is reduced to finding edge disjoints paths for the logical links in the physical topology, which is a well known non-deterministic polynomial time (NP)-complete problem [5]. For general logical topologies, it may not be necessary to find edge disjoint paths for all the logical links in the physical topology. Mapping a subset of the logical links in a disjoint manner would be sufficient but again problem of finding disjoint paths is NP-complete, if number of logical links to be mapped is more than 2 for arbitrary topologies [6].

## II. LITERATURE REVIEW

The problem of finding edge disjoint path bears a very close resemblance to the *multicommodity flow problem* (MCF). The multicommodity flow problem involves simultaneously routing several different commodities in a network, from their sources to their respective destinations, such that the maximum flow on each edge is not greater than its capacity while meeting certain objective. In case of *minimum cost maximum flow multicommodity flow* (MCMF), the aim is to maximize the flows while minimizing the cost of the routing the flows. When the logical topology is cycle, MCMF formulation can be directly used to find survivable mapping by setting the capacities, demands and costs to 1 and requiring integer flows only. Then a survivable mapping is possible only if a feasible solution to the MCMF exists. For arbitrary graphs, a feasible solution to the integer MCMF implies that a survivable mapping exists but an infeasible solution does not imply that such mapping does not exist. This is due to the fact that we are interested in finding disjoint paths for some rather than all logical edges.

In [5], Modiano *et. al.* provide an integer linear program (ILP) for arbitrary logical topologies, based on the observation that a single physical link failure can disconnect the logical topology only if it carries the entire cut set of the logical topology. The formulation adds a survivability constraint to the MCMF. If the formulation has a feasible solution, a survivable mapping exists for the logical topology and an infeasible solution implies that no such mapping exists. [5] also notes that when the logical topology is a cycle (ring), the survivability constraint in the above formulation can be removed. [5] also proves that the above formulation is NP-complete, even for a ring logical topology.

[7] also considers the problem of survivable mappings under the protection interoperable design paradigm. [7] defines a *protected group with protection level k* as a group of logical links that can support up to *k* logical links failures. Such groups are assumed to have the ability to reroute traffic on broken links as long as the number of broken links is *k* or less. [7] imposes three constraints (i) *capacity constraint* requires that a mapping should respect the capacity constraints of the physical link (ii) the *bottleneck constraint* demands that one physical link/node failure should not cause more than *k* logical link failures (iii) the *connectivity constraint* implies that the logical topology must remain connected for any single link/node failure in the physical topology. [7] also notes that the problem is NP-complete.

Since the problem of finding disjoint paths is NP-complete in general, efficient algorithms for finding such paths are unlikely in their full generality. Therefore, a significant amount of literature is dedicated to finding good heuristics. [5] proposes a heuristic that considers only those cuts that prevent a single node from being isolated. [7] proposes a heuristic based on tabu search meta-heuristic, called *protection interoperability for WDM* (PIW). The PIW starts by arbitrary routing logical links, at each iteration PIW slightly modifies the routes of a logical links to obtain a new mapping. The routing at each step is the best network mapping that has not been visited yet. After a given number of iterations, if the mapping does not improve, the algorithm terminates.

[8] also provides a solution to the problem of finding disjoint paths. Given a set of *K* source-destination (*s*, *t*) pairs, the objective is to determine disjoints paths among as many source-destination pairs as possible. The same problem is considered in [9] and an iterative greedy algorithm called *bounded greedy algorithm* (BGA) is provided. BGA starts off with an empty solution *S*, picks an (*s*, *t*) from *K* and routes it using a shortest path *P* of length *L* or less. After routing, the edges along *P* are removed from the graph and the algorithm proceeds by picking the next (*s*, *t*) pair in *K*. An extension of BGA is proposed in [8] called *multi-start greedy algorithm* (MSGA). In an iteration of MSGA, $S_i$ is the solution under consideration and $S_{best}$ is the best solution achieved so far. Also, in each iteration of MSGA, a random permutation of *K* is used. The main contribution of [8] is the development of an algorithm based on *ant colony optimization* (ACO). A solution *S* is constructed by assigning an ant to each (*s*, *t*) pair, which finds a path for the assigned (*s*, *t*) pair. Initially the paths in *S* may not be mutually disjoint, edge disjoint paths are then obtained by iteratively removing the path which has the most edges in common with other paths in *S*. The algorithm continues until edge disjoint paths are found or the termination condition is met.

## III. KURANT THIRAN APPROACH

In [10], the authors provide a unique approach to find survivable mapping for a general logical topology by mapping only a subset of the logical links. [10] accomplishes this by introducing the concept of *piecewise survivability*, which proceeds by recursively finding survivable mappings for the *pieces* of the logical topology rather than for the entire logical topology. The algorithm is based on the following important property. Suppose the given logical topology is 2-edge connected. If a subgraph, $G_1$ is routable in a survivable manner and a subgraph $G_2$ in the contracted subgraph is also routable in a survivable manner, then the subgraph of the logical topology containing the links of $G_1$ and $G_2$ is also routable in a survivable manner [11]. This approach greatly simplifies the process of finding survivable mappings for fairly large networks. The solution is applicable to arbitrary logical and physical topologies and can be applied to find link survivable mapping as well as to find node survivable mappings.

The algorithm starts with the entire logical topology, and then tries to find a subgraph for which a survivable mapping can be found. If such a subgraph exists, the algorithm *contracts* this subgraph by collapsing the edges and merging the nodes of the subgraph to create a contracted logical topology. After contracting the subgraph, the algorithm proceeds by finding another subgraph that can be mapped in a survivable manner. The algorithm terminates, if no such subgraph is found or the logical topology is reduced to a single node. If the algorithm terminates unsuccessfully, it returns a *piecewise link/node-survivable mapping*, which consists of survivable *pieces* and the remaining contracted logical topology.

However, in this approach, one may be required to consider a large number of subgraphs until a survivable subgraph is found. [10] also considers the special case when the selected subgraph is always a cycle. If the subgraph is restricted to a cycle at each iteration, then an undirected complete graph has $\frac{1}{2}\sum_{i=3}^{|N|}\binom{|N|}{i}(i-1)!$ cycles, a number which grows faster with $|N|$ than the exponential $2^{|N|}$ [12]. This is a computationally expensive task. So an implementation of this algorithm would be limited to considering a small number of subgraphs, and may terminate unsuccessfully even though a survivable mapping was possible.

Due to the NP-complete nature of the disjoint paths problem, [10] provides a simple heuristic to find mappings for logical links when the subgraph is always a cycle (ring). The heuristic is named SMART, *survivable mapping algorithm by ring trimming*. The heuristic uses a variant of shortest path algorithm, therefore does a local search. The algorithm assigns a weight of 1 to each physical link and finds a shortest path for each logical link. If the paths obtained are edge disjoint, the algorithm proceeds by contracting the subgraph under consideration and picking a new subgraph. If some or all the paths are not edge disjoint, the weights assigned to the shared physical edges are incremented by 1 and the process is repeated a predetermined number of times. In their more recent work in [13], the heuristic is referred to as SMART-H and the framework as SMART.

Most approaches in literature consider the entire logical topology while designing survivable networks. [10] provides a different approach by considering only pieces of the logical topology. Nonetheless, it has certain drawbacks due to the NP-completeness of the problem. One drawback is the number of subgraphs which it may have to consider before declaring that a survivable mapping of the logical topology in the physical topology does not exist. The other drawback is the heuristic used to find mappings may terminate unsuccessfully even though survivable mappings may exist. The aim of this paper is to explore more sophisticated methods for picking subgraphs and finding survivable mappings.

In this paper, we consider the special case when the selected subgraph is always a cycle. If the subgraph is a cycle, the problem of finding survivable mappings is reduced to the problem of finding edge disjoints paths. In other words, if paths assigned to all the links on the cycle are mutually edge disjoint, the mapping is survivable. This is easy to verify in contrast to the case when the subgraph is an arbitrary subgraph. In such a case, the existence of survivability must be checked by removing a physical link and the logical links using this physical link, and conducting a test whether the logical topology remains connected after their removal. It is also easy to pick a cycle as a subgraph using a shortest path algorithm.

A cycle is the minimum two-connected graph, therefore, if an arbitrary subgraph is picked which can be mapped in survivable manner, it can be reduced to a set of links that form at least one cycle and their mappings are edge disjoint. Therefore, just considering cycles as subgraphs should be sufficient to determine whether a survivable mapping for a logical topology exists in a given physical topology.

As mentioned earlier, a large number of cycles may exist in a given graph and all of them must be considered before declaring that a solution does not exist. In the worst case we may be forced to find a Hamiltonian cycle, which is a well known NP-complete problem. Therefore, in this paper we propose an enhancement that picks a cycle and attempts to map it. If the chosen cycle can be mapped in a disjoint manner, the algorithm proceeds by contracting this subgraph and picking a new cycle. However, if the chosen cycle cannot be mapped in a survivable manner, the cycle is modified to make it survivable. In the next section, we describe an approach that pick a cycle and tries to modify it, if the cycle is not initially survivable. Section IV also provides a mapping algorithm that does a more rigorous search to find disjoint mapping.

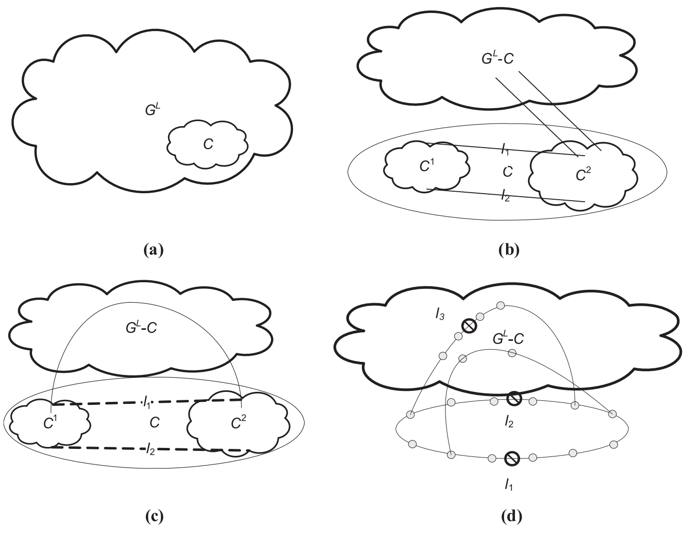


Fig. 3. Illustration of enhancement 1: (a) Logical topology $G^L$ and a cycle $C$, (b) survivable mapping not found for $C$ ($C$ is split into two components and a component gets isolated.), (c) survivable mapping not found for $C$ ($C$ is split into two components but the components are not isolated.), and (d) modifying the cycle $C$ using step 1.

## IV. A ROBUST SURVIVABLE TOPOLOGY DESIGN APPROACH

In this section we describe a new approach to design survivable networks based on the framework provided by Kurant-Thiran as described in Section III. The new approach called *modify and map* (MM) provides a more systematic approach to finding subgraphs (cycles) to map in the logical topology (enhancement # 1) and a mapping algorithm based on the concept of randomized rounding utilizing a fractional multicommodity flow approximation algorithm (enhancement # 2).

### *Enhancement # 1: Subgraph Selection*

Assume that we are given a physical topology $G$ ($N$, $E$) and a logical topology $G^L$ ($N^L$, $E^L$), where $N$ is the set of physical nodes, $E$ is the set of physical edges, $N^L$ is the set of logical nodes such that $N^L \subseteq N$, and $E^L$ is the set of logical links. Also assume that both physical and logical topologies are at least 2-connected. The algorithm proceeds by picking a subgraph $C$ in the form of a cycle in the logical topology and attempts to map it in the physical topology. If $C$ can be mapped in a disjoint manner, the logical topology is contracted by collapsing the edges and merging the nodes in $C$. However, if $C$ cannot be mapped in a disjoint manner, there is a set of links, such that they share some physical link/links. In this case $C$ will be split into $|\psi|$ components, $C^1, C^2, \cdots, C^\psi$ . Fig. 3 shows the case when the cycle $C$ is split into 2 components $C^1$, and $C^2$, i.e., $|\psi|$ = 2. When $C$ is split into two or more components and a component gets isolated i.e. the component is no longer connected to any other component or the graph $G^L - C$, then it is not possible for the entire logical topology to be survivable without finding a cycle, which contains the links that connect the isolated component to the other components. This scenario is shown in Fig. 3(b). In this case, it is necessary to do an exhaustive search to verify that no such cycle exists. Fig. 3(c) shows another possibility. In this case none of the components get isolated i.e. there exists a path or paths in $G^L - C$ that connect one component to the other component. In that case we proceed as follows:

***Step 1***: Pick a link $l_i \in \psi \ (1 \leq i \leq \psi)$, and find the two components $C^1$ and $C^2$ it connects. Let $n_1$ be the set of degree 3 nodes in component $C^1$ and $n_2$ be the set of such nodes in the component $C^2$. Now pick a node $n' \in n_1$ and a node $n'' \in n_2$. Let $P'$ be the path between $n'$ and $n''$ on the cycle $C$. After finding such nodes, remove all the edges belonging to the cycle $C$ in the logical topology and find a shortest path $P$ between $n'$ and $n''$. The edges are then restored and a cycle is formed by concatenating $P'$ and $P$ and an attempt is made to map this cycle. If this cycle can be mapped in disjoint manner, $\psi$ is updated by removing $l_i$ and $C$ is updated by collapsing the edges and merging the nodes in $P'$ and $P$. Otherwise, this procedure is repeated by pairing the nodes from $n_1$ with nodes from $n_2$. If for a link $l_i \in \psi$, no such cycle can be found, we proceed with $l_{i+1} \in \psi$.

Suppose that survivable mappings can be found for $|\psi| - 1$ links, then it can be shown that the edges in the $|\psi| - 1$ cycles along with those in $C$ form a graph which will connected for any single physical link failure. So we contract these edges and proceed with the remaining edges in the logical topology.

***Step 2***: If we are unable to find survivable cycles for $|\psi| - 1$ links, there are some for which survivable cycles could not be found. Let $C'$ be the cycle after step 1. Now we pick a pair of nodes $(x, y)$ with degree $\geq 3$ in $C'$ and find a cycle, which includes the path on $C'$ between $x$ and $y$ and a path between $x$ and $y$ that does not include any edge from $C'$. If this cycle can be mapped in a survivable manner, the edges in this cycle are collapsed. This procedure is repeated until all the edges have been collapsed to create a single node or until all the nodes with degree $\geq 3$ have been processed.

***Step 3***: Let $C''$ be cycle after step 1 and 2. If $C''$ is not a single node, we apply the mapping algorithm to $C''$. If the mapping is survivable, the edges are contracted and nodes are merged. However, if $C''$ cannot be mapped in a survivable manner, then the unsurvivable mapping is accepted and the logical topology is contracted but a flag is set to indicate that an unsurvivable mapping was accepted. This requirement guarantees the termination of the algorithm. The algorithm then proceeds by selecting a new cycle $C$.

***Step 4***: If the flag was set, after the termination of the algorithm, logical edges that were not mapped earlier are mapped in an arbitrary manner and a test is conducted to see if the mapping of the entire logical topology is survivable. Since the problem of finding disjoint mappings is NP-complete, any algorithm other than the *Integer Linear Program* (ILP) formulation cannot guarantee that if a solution is not found then a solution does not exist. The reasoning for accepting unsurvivable mapping for the cycle and continuing is that the cycles selected in subsequent iterations may aid in making the unsurvivable subgraphs survivable. The entire algorithm is shown in Figs. 4, 5, 6, and 7.

```
INPUT: An at least 2-connected, physical topology $G(N,E)$, a logical
topology $G^L$ ($N^L$, $E^L$)
OUTPUT: One link survivable mapping $M$ of $G^L$ in $G$
While the termination condition is NOT met do
  $C \leftarrow$ a short cycle in $G^L$
  Call the mapping procedure
  If mapping procedure returns a survivable mapping for
  $C$ then
    Contract $G^L$ by collapsing the edges and merging
    nodes in $C$
  Else
    call modify and map
  End if
End While
If Survivable = false then
  Map unmapped edges
  For all physical edges do
    Remove the physical edge from the physical
    topology and remove the logical links that use
    this edge from the logical topology.
    If the logical topology remains connected the
      Continue
    Else
      Logical topology cannot be mapped in survivable
      manner
      END
    End If
  End For
End If
```

Fig. 4. Main routine.

### *Enhancement # 2: A Rigorous Mapping Algorithm*

One key assumption in the above algorithm is that there exists a method to find disjoint mappings for a given set of links. As mentioned earlier, the problem of finding mutually disjoint paths for a given set of links is NP-complete in general. The problem can be formulated as an ILP and solved exactly. Since ILP exhibits excessive runtimes, it may not be suitable for large networks. Due to the fundamental role played by disjoint paths in designing telecommunication networks, a significant amount of literature is dedicated to solving this problem without solving the ILP formulation. Most of the proposed approaches assign weights to the physical links which are then used to find paths, usually using a shortest path algorithm. After finding a path the weights are usually updated for the links in the path. The process is repeated until disjoints paths are found or a termination condition is met. Such approaches work well in practice but an unsuccessful termination does not necessarily imply that disjoint mappings are not possible.

In this paper we propose a variation of the randomized rounding technique proposed by Raghavan and Thompson to find disjoint paths in an undirected network [14]. The algorithm transforms the optimal solution of a relaxed 0-1 integer formulation into a "*provably good*" solution to the original 0-1 formulation [14]. The algorithm relaxes the integer constraint of the MCF problem to obtain fractional solutions, which are then used to obtain integer solutions. [14], however, does not provide a method to solve the relaxed problem and assumes fractional flows are available to the algorithm. Algorithms in [15] and [16] provide fast algorithms to solve the fractional MCF problem in polynomial times, which can then be used to get integer solutions by applying the approach in [14].

To get fractional solutions we use a modified version of the multicommodity flow approximation algorithm in [15]. To use the algorithm in [15] and [16], the links belonging to the subgraph (a cycle $C$) are viewed as commodities $K$ =

```
INPUT: An at least 2-connected, physical topology $G(N, E)$, a logical topology $G^L(N^L, E^L)$, a cycle $C$
OUTPUT: Return success if the mapping is survivable
Procedure modify and map
$\psi \leftarrow$ logical links belonging to $C$ for which disjoint mappings could not be found
For all logical link $l_i \in \psi$ do
    $C_i^1 \leftarrow$ a component that $l_i$ connects
    $C_i^2 \leftarrow$ other component that $l_i$ connects
    $n_1 \leftarrow$ list of nodes with degree > 2 in $C_i^1$
    $n_2 \leftarrow$ list of nodes with degree > 2 in $C_i^2$
    If $n_1 = \phi$ OR $n_2 = \phi$ then
        continue
    End If
    For all $n' \in n_1$ do
        For all $n'' \in n_2$ do
            $P' \leftarrow$ path between $n'$ and $n''$ on the cycle $C$
            $P \leftarrow$ path between $n'$ and $n''$ in $G^L$ - $C$
            $C_i \leftarrow P \cup P'$
            If mapping procedure returns a survivable
            mapping for $C_i$ then
                Contract $G^L$ by collapsing edges and merging
                nodes in $C_i$
                Contract $C$ by collapsing edges and merging
                nodes in $P'$
                Remove $l_i$ from $\psi$
                Continue with the next $l_i$ in $\psi$
            End If
        End For
    End For
    If $|\psi|$ < 2 then
        Contract $G^L$ by collapsing remaining edges
        and merging nodes in $C_i$
        Return Success
    End If
End For
$D$ = list of nodes with degree > 2 on cycle $C'$
While $|D|$ > 1 do
    $\gamma_1 \leftarrow$ remove a node from $D$
    For all nodes $\gamma_2 \in D$ do
        $P' \leftarrow$ path between $\gamma_1$ and
        $\gamma_2$ on the cycle $C'$
        $P \leftarrow$ a short path between $\gamma_1$ and $\gamma_2$ in $G^L$ - $C'$
        $C_i' \leftarrow P \cup P'$
        Call the mapping procedure
        If mapping procedure returns a survivable mapping
        for $C_i'$ then
            Contract $G^L$ by collapsing edges and merging
            nodes in $C_i'$
            Contract $C'$ by collapsing the edges and merging
            the nodes in $P'$
        End if
    End For
End While
If subgraph is reduced to a single node then
    Return *success*
Else
    If mapping procedure returns a survivable mapping
    for $C''$ then
        Contract $G^L$ by collapsing the edges and merging nodes in $C''$
        Return *success*
    Else
        Contract $G^L$ by collapsing the edges and merging nodes in $C''$
        Survivable = *false*
        Return *success*
    End if
End If
```

Fig. 5. Modify and map.

$(s_1, t_1), (s_2, t_2), \ldots\ldots, (s_k, t_k)$, where $k$ is the number of links in the cycle and $s_i$ and $t_i$ are the end nodes of a logical link $l \in C$. All the physical links are assigned a capacity of 1 ($u_{ij} = 1$) and the objective is to find flows $f_i$ from $s_i$ to $t_i$ without violating the capacity constraints of the physical edges.

```
INPUT: A network $G$, capacities $u(e)$, commodity pairs $K = (s_k, t_k)$, $1 \le k \le |K|$, and accuracy $\epsilon$
OUTPUT: Best mapping possible for pairs in $K$, solve fractional multicommodity flow problem.
Procedure mapping algorithm:
Initialize $l(e) = \delta, \forall e, x \equiv 0$
$K' \leftarrow K$
Best Mapping $M_{Best} \leftarrow \phi$
$|M_{Best}| \leftarrow \alpha$  /*number of non-disjoint links*/
Commodities $\leftarrow |K'|$
While there is a path $P$ between a $(s_k, t_k) \in K'$ such that $l(P)$ < 1 do
    Mapping $M \leftarrow \phi$
    For all $(s_k, t_k) \in K'$
        Find the shortest paths $P$ using $l$
        $M \leftarrow M \cup P$
        If $l(P)$ < 1 then
            $u \leftarrow min_{e \in P} u(e)$
            $x(P) \leftarrow x(P) + u$
            $\forall e \in P,\ l(e) \leftarrow l(e)\left(1 + \frac{\epsilon u}{u(e)}\right)$
        else
            $K' \leftarrow \{K' - (s_k, t_k)\}$
        End If
    End For
    If Commodities = $|K'|$ and $M$ is edge disjoint then
        Return *Success*
    Else If commodities = $|K'|$ then
        If number of non-disjoint links in $M < |M_{Best}|$ then
            $M_{Best} \leftarrow M$
        End If
    End If
    permute $K'$
End While
Call Randomized Rounding
```

Fig. 6. Fractional multicommodity flow approximation algorithm.

Algorithms in [15] and [16] are based on the combinatorial knowledge of the dual of the path-flow multicommodity flow formulation to provide $\epsilon$-approximate solutions. The formulation is given below:

$$\max \sum_{p \in P} x(P)$$

$$\text{Capacity constraints:} \sum_{P:(i,j) \in P} x(P) \le u_{ij},\ \forall (i,j) \in P$$

$$\text{Flow constraints: } x(P) \ge 0,\ \forall P. \quad (1)$$

Here, $x(P)$ is the amount of flow sent along a path $P$ (initially 0), then the dual of above formulation can be formed by assigning lengths to the edges of the graph. The length of an edge, $l_{ij}$, is related to the amount of flow it carries and the maximum flow computations are done only on this length function. The objective is to minimize $\sum_{(i,j) \in E} u_{ij}.l_{ij}$, such that the length of the any shortest path between any $(s_i, t_i)$ pair is at least 1, for all $(s_i, t_i) \in K$. The length of each edge is set to $\delta = \frac{(1+\epsilon)}{((1+\epsilon)L)^{\frac{1}{\epsilon}}}$, where $L$ is the length of the longest path between any$(s_i, t_i) \in K$ and $\epsilon$ is the desired accuracy. The algorithm proceeds by finding shortest paths $P_i$ for all $(s_i, t_i) \in K$ using the length function and picking the shortest path $P \in P_i$, the flows and lengths of the edges on $P$ are then updated.

The algorithm in [15] always picks the commodity with the shortest path among all the commodities to push flow, which

```
INPUT: A network $G$, capacities $u(e)$, flow matrix $f$, commodity pairs
$K = (s_k, t_k)$, $1 \leq k \leq |K|$, the Best Mapping from mapping algo-
rithm $M_{Best}$
OUTPUT: Disjoint paths
Procedure randomized rounding:
$\pi_k \leftarrow$ set of paths for commodity $k$
For all commodities $k$ in $K$ do
  $G_k(V, E_k) \leftarrow$ A directed graph with $E_k \subseteq E$ such
  that $e \in E_k$ iff $e$ has a non-zero fractional flow in the
  solution to MMCF for commodity $k$
  While there is non-zero flow leaving $S_i$ do
    $P \leftarrow$ a directed path in $G_k$ for commodity $k$, using
    Dijikstra shortest path algorithm
    $\pi_k \leftarrow \pi_k \cup P$
    $f_m \leftarrow$ flow along the bottleneck link (minimum
     flow)in path $P$
    Add the path $P$ and the flow $f_m$ to $\pi_k$
    Reduce flows for all $e \in P$ for commodity $k$ by $f_m$
    Remove the edges from $G_k$ for commodity $k$ for
    which $f_e = 0$
  End While
End For
While no of rounds is less than threshold do
  Mapping $M \leftarrow \phi$
  For all commodities $k$ in $K$ do
    Cast a die with number of faces = $|\pi_k|$ with the
    face probabilities equal to flows for the paths in $\pi_k$
    Roll the dice and assign the path $P$ to commodity $k$ whose
    face comes up
    $M \leftarrow M \cup P$
  End For
  If $M$ is edge disjoint then
    Return Success
  Else If number of non-disjoint links in $M$ < $|M_{Best}|$
  then
    $M_{Best} \leftarrow M$
  End if
End While
```

Fig. 7. Randomized rounding.

may starve some commodities i.e. they may not get any flow. This is undesirable when finding disjoint paths using randomized rounding. To circumvent this problem we use a modified version of the above algorithm. Instead of picking the commodity with the shortest path, we go through the commodities in a round robin manner to provide each a chance to send flow. To introduce further fairness, the commodities are permuted after each iteration. The modified algorithm is given in Figs. 6 and 7.

## V. SIMULATION STUDY

To study and compare the behavior of the proposed algorithm, and SMART-H, they were implemented using library of efficient data type and algorithms (LEDA) [17] and VC++ 8.0. For simulation physical and logical topologies with varying number of nodes and edges were generated. Physical topologies were regular topologies with 500 and 1000 nodes ($|N|$) and degrees 6 and 8. The regular topologies were constructed using a procedure originally given by Harary and described in [18]. Logical topologies were random topologies generated using LEDA. The logical nodes ($|N^L|$) were a random subset of physical nodes ($0.8 \times |N|$). The number of logical links ($E_L$) were 1.5 and 2.0 times the logical nodes. After generating the logical topologies, they were admitted for further processing only if they were at least two edge connected. SMART-H and the proposed algorithm were then applied to each logical-physical topology pair.

Table 1. Performance comparison of SMART-H and MM.

| $|N|$ =500, $|N^L|$ =400 | | | | | |
|---|---|---|---|---|---|
| | Degree | $|E^L|$ | No. of Survivable Mappings | Ave. subgraph size (edges) | Ave Time per Topology (min) |
| MM | 6 | 600 | 884 | 4 | 5.074 |
| SMART-H | 6 | 600 | 800 | 4 | 4.286 |
| MM | 8 | 600 | 1056 | 3 | 5.735 |
| SMART-H | 8 | 600 | 932 | 3 | 5.081 |
| MM | 6 | 800 | 1061 | 3 | 3.895 |
| SMART-H | 6 | 800 | 1004 | 3 | 3.027 |
| MM | 8 | 800 | 1101 | 3 | 3.344 |
| SMART-H | 8 | 800 | 1020 | 3 | 2.875 |

Table 2. Performance comparison of SMART-H and MM.

| $|N| = 1000$, $|N^L| = 800$, Degree=8, $|E^L| = 1600$ | | | | |
|---|---|---|---|---|
| | No. of Survivable Mappings | Ave. Subgraph Size edges | Ave Time per Topology (min) | Ave. No of Shortest path applications |
| MM | 1104 | 3 | 12.88 | 3598 |
| SMART-H | 991 | 3 | 9.286 | 3085 |

The total number of such logical-physical topology pair was 1200 for each case (40 physical and 30 logical topologies).

To make the SMART-H practical, if a disjoint mapping could not be obtained after applying the mapping method 100 times for $|N|$ = 500 and 200 for $|N| = 1000$, a new subgraph was selected and SMART-H terminated after unsuccessfully examining 100 subgraphs for $|N| = 500$ and 200 for $|N| = 1000$. For the proposed algorithm, $\epsilon$ was 0.15 and the number of randomized rounding rounds were set 50. To find a subgraph, two nodes were randomly picked and two link disjoint paths were found between these nodes. The two paths were then concatenated to get the subgraph.

The results are summarized in Tables 1 and 2. It can be seen that the proposed algorithm can map more logical topologies in survivable manner then SMART-H (Table 1). As one would expect, the proposed algorithm does more exploration and so performs more number of shortest path computations, leading to increased execution times (Table 2). But the increased execution times is compensated by the increased number of survivable mappings generated.

To understand why SMART-H finds fewer survivable mappings, first recall that when a mapping of a subgraph is not survivable, only the weights of the shared physical edges are updated in SMART-H. This may modify the paths obtained in the next iteration only slightly. This means that the mapping generated in the next iteration may still not be survivable. Since the number of unsuccessful attempts is set to a predetermined value, SMART-H may not be examining as many mappings as one would have expected. Since the subgraph selection is random, therefore, it may take several unsuccessful attempts before a survivable subgraph can be found. Also SMART-H does not have a mechanism to keep track of the subgraphs that have already been considered, it may pick an unsurvivable subgraph several times, thereby again not exploring as many subgraphs as one would have expected.

To explain why the proposed algorithm takes more execution time, note that if a subgraph is not survivable, enhancement # 1 is applied, which may map the chosen subgraph and/or some subgraphs surrounding it. However, finding such subgraphs requires extra overhead. Also, keeping track of the best mapping is another overhead involved. The increased time required by the overheads is alleviated to some extent by the efficient randomized rounding approach. Also, in the randomized rounding approach the weights (length) of all the links that are part of the current mapping are updated, thereby reducing the possibility of an unsuccessful mapping being considered more than once.

It can also be seen in Tables 1 and 2 when the topologies (logical or physical) are sparse, fewer logical topologies can be mapped in survivable manner. However, making the topologies dense increases the number of survivable mapping. This is why dense logical topology implies that more subgraphs are available. Also if the physical topology is dense a larger number of paths will be available.

## VI. CONCLUSION

In this paper we have proposed an algorithm based on the framework developed in [10] that starts by picking a subgraph in the logical topology and attempts to map the links in the subgraph in disjoint manner in the physical topology. This greatly simplifies the process of finding survivable mapping. The proposed algorithm uses the concept of randomized rounding discussed in [14] to find disjoint paths, and is able to achieve higher success rate than SMART-H.

## ACKNOWLEDGMENTS

The work of K. Thulasiraman has been supported in part by NSF ECS grants ANI-0312435 and ECS 04-26831 and the work of G. Xue has been supported by NSF ITR grant ANI-0312635. We would also like to thank the referees for their comments and suggestions to improve the quality of this paper.

## REFERENCES

[1] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications an approach to high bandwidth optical WDM," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171–1182, July 1992.

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

[3] R. Ramaswami and K. Sivarajan, *Optical Networks: A Practical Perspective*. Morgan Kaufmann, 1998.

[4] A. Fumagalli and L. Valcarenghi, "IP restoration vs. WDM protection: Is there an optimal choice?," *IEEE Network*, vol. 14, no. 6, pp. 34–41, Nov./Dec. 2000.

[5] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: A new approach to the design of WDM-based networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 800–809, May 2002.

[6] T. Tholey, "Solving the 2-disjoint paths problem in nearly linear time," *Theory of Computing Systems*, vol. 31, no. 1, pp. 51–78, Feb. 2006.

[7] O. Crochat, J. L. Boudec, and O. Gerstel, "Protection interoperability for WDM optical networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 384–395, June 2000.

[8] M. Blesa and C. Blum, "Ant colony optimization for the maximum edge-disjoint paths problem," *Springer-Verlag Berlin Heidelberg*, vol. 3005/2004, pp. 160–169, 2004.

[9] *Approximation Algorithms for Disjoint Paths Problems*, Thesis by Jon Michael Kleinberg.

[10] M. Kurant and P. Thiran, "On survivable routing of mesh topologies in IP-over-WDM networks," in *Proc. IEEE INFOCOM 2005*, vol. 2, Mar. 2005, pp. 1106–1116.

[11] M. Javed, K. Thulasiraman, M. Gaines, and G. Xue, "Survivability aware routing of logical topologies: On Thiran-Kurant approach, evaluation and enhancements," in *Proc. IEEE Globecom 2006*, Nov. 2006, pp. 1–6.

[12] P. Mateti and N. Deo, "On algorithms for enumerating all circuits of a graph," *SIAM Journal on Computing*, vol 5, no. 1, pp. 90–99, Mar. 1976.

[13] M. Kurant and P. Thiran, "Survivable routing of mesh topologies in IP-over-WDM networks by recursive graph contraction," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, pp. 922–933, June 2007.

[14] P. Raghavan and C. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–375, Dec. 1987.

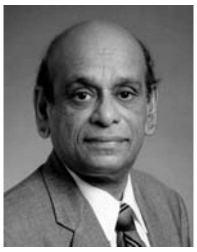
[15] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. IEEE Symposium on Foundations of Computer Science 1998*, Nov. 1998, pp. 300–309.

[16] L. Fleischer, "Approximation fractional multicommodity flow independent of the number of commodities," *SIAM J. Discrete Math*, vol. 13, no. 4, pp. 505–520, 2000.

[17] K. Mehlhorn and S. Näher, *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University, 1999.

[18] K. Thulasiraman and M. Swamy, *GRAPHS: Theory and Algorithms*. Wiley-Inter-science, 1992.

**Muhammad Javed** received the Bachelor's degree (1995) in Civil Engineering from University of Engineering and Technology, Pakistan, MBA (1997) from Oklahoma City University, Oklahoma and Master's degree (2000) in Computer Science from the University of Oklahoma, Oklahoma. He is currently working towards his Ph.D. at University of Oklahoma, Oklahoma. His research interests include network computing, fault tolerant system, and graph theory.



**Krishnaiyan Thulasiraman** received the Bachelor's degree (1963) and Master's degree (1965) in Electrical Engineering from the University of Madras, India, and the Ph.D. degree (1968) in Electrical Engineering from IIT, Madras, India. He holds the Hitachi Chair and is Professor in the School of Computer Science at the University of Oklahoma, Norman, where he has been since 1994. Prior to joining the University of Oklahoma, he was professor (1981-1994) and chair (1993-1994) of the ECE Department in Concordia University, Montreal. He was on the faculty in the EE and CS departments of the IITM during 1965-1981. His research interests have been in graph theory, combinatorial optimization, algorithms and applications in a variety of areas in CS and EE. He has published more than 100 papers in archival journals, coauthored with M. N. S. Swamy two text books "Graphs, Networks, and Algorithms" (1981) and "Graphs: Theory and Algorithms" (1992), both published by Wiley Inter-Science. He has received several awards and honors that include the 2006 IEEE Circuits and Systems Society Technical Achievement Award. He has been very actively professionally in the IEEE Circuits and Systems and other societies.

**Guoliang (Larry) Xue** is a Professor in the School of Computing and Informatics at Arizona State University (ASU). He earned a Ph.D. (1991) in Computer Science from the University of Minnesota (Minneapolis, USA), an MS (1984) in Operations Research and a BS (1981) in Mathematics, both from Qufu Teachers University (Qufu, China). Before joining ASU in 2001, he had worked at Qufu Teachers University as a Lecturer (1984-87), the Army High Performance Computing Research Center as a Postdoctoral Fellow (1991-93), and The University of Vermont as an Assistant/Associate Professor (1993-2001). He is a member of ACM and a senior member of IEEE.

His interests include Quality of Service routing, resource allocation, survivability and security issues in networking (both wireless and wireline), with a strong flavor of optimization and algorithmics. He has published over 160 papers in these areas.