Simple Pre-Provisioning Scheme to Enable Fast Restoration

Mansoor Alicherry and Randeep Bhatia

Abstract—**Supporting fast restoration for general mesh topologies with minimal network over-build is a technically challenging problem. Traditionally, ring-based SONET networks have offered close to 50 ms restoration at the cost of requiring 100% over-build. Recently, fast (local) reroute has gained momentum in the context of MPLS networks. Fast reroute, when combined with pre-provisioning of protection capacities and bypass tunnels, enables faster restoration times in mesh networks. Pre-provisioning has the additional advantage of greatly simplifying network routing and signaling. Thus, even for protected connections, online routing can now be oblivious to the offered protection, and may only involve single shortest path computations.**

In this paper, we are interested in the problem of reserving the least amount of the network capacity for protection, while guaranteeing fast (local) reroute-based restoration for all the supported connections. We show that the problem is NP-complete, and we present efficient approximation algorithms for the problem. The solution output by our algorithms is guaranteed to use at most twice the protection capacity, compared to any optimal solution. These guarantees are provided even when the protection is for multiple link failures. In addition, the total amount of protection capacity reserved by these algorithms is just a small fraction of the amount reserved by existing ring-based schemes (e.g., SONET), especially on dense networks. The presented algorithms are computationally efficient, and can even be implemented on the network elements. Our simulation, on some standard core networks, show that our algorithms work well in practice as well.

Index Terms—**Approximation algorithms, fast shared restoration, local reroute, MPLS, optical, pre-provisioning.**

I. Introduction

Modern backbone and transport networks are highly complex networks that strive to carry services with QoS guarantees. These networks support general topologies and dynamic routing of bandwidth guaranteed connections, yet at the same time they aim to provide fast recovery from network failures. Traditionally, ring-based SONET networks have offered close to 50 ms restoration to bandwidth guaranteed services, using pre-reserved spare protection capacity and pre-planned protection paths. Pre-planning protection in rings has been especially attractive, because of the availability of exactly one backup path between any two nodes, leading to very simple and fast automatic protection switching mechanisms. However, in ring-based SONET networks these advantages come at the cost of reserving at least half the total capacity for protection, thus requiring 100% redundancy.

Recently, mesh-based networks have received much attention due to the increased flexibility they provide in routing connections, thus leading to more efficient utilization of network resources. Also mesh networks are appealing due to the high degree of protection capacity sharing that is possible in these networks, offering the promise of fast restoration times of ring-based SONET networks for just a small fraction of the total capacity, reserved for protection. However, it has remained a challenging problem to design such efficient protection schemes for mesh networks. In general most protection schemes, including those for SONET and ring-based schemes, have been designed to protect against a single link failure. It is also a challenging problem to design efficient protection schemes that protect against multiple link failures for mesh networks.

Recently, a fast (local) reroute [9] approach to restoration has gained momentum in the context of Multi-Protocol-Label-Switching (MPLS) [5] networks. The MPLS fast or local reroute supports a local repair capability where upon a node or link failure the first node upstream from the failure reroutes the effected Label Switch Paths (LSP) onto bypass (backup) tunnels with equivalent guaranteed bandwidths, thereby achieving faster restoration times. The MPLS fast reroute mechanism allows for bandwidth sharing between bypass tunnels protecting independent resources, thus resulting in efficient capacity utilization.

Two different techniques for local protection in MPLS networks have been proposed [22]. The one-to-one backup technique [1], [13], [15] creates bypass LSPs for each protected service carrying LSP, at each potential point (link or node) of local repair. The facility backup technique [29] creates a bypass tunnel to protect a potential failure point (link or node), such that by taking advantage of the MPLS label stacking mechanism, a collection of LSPs with similar backup constraints can be jointly rerouted, over a single bypass tunnel. In general, the one-to-one backup technique does not scale very well with the number of supported protected LSPs, since the number of bypass tunnels can quickly become very large, not to mention the enormous load on signaling and routing to support these extra tunnels. In addition, for implementing the one-to-one backup technique, either extensive routing extensions are needed to propagate the set of bypass LSPs and their attribute information [1], resulting in heavy load on the control plane, or the amount of achievable sharing of protection capacity is sacrificed, by limiting the amount of state that is propagated in the routing updates [13], thus requiring large amounts of spare capacity for protection.

Manuscript received October 25, 2004; revised August 16, 2005, and January 5, 2006; approved by IEEE/ACM Transactions on Networking Editor A. Somani. A preliminary version of this paper appeared in IEEE INFOCOM 2004, Hong Kong.

The authors are with Bell Labs, Lucent Technologies, Murray Hill, NJ 07974 USA (e-mail: mansoor@research.bell-labs.com).

Digital Object Identifier 10.1109/TNET.2007.892844

The facility backup technique is free from many of the drawbacks of the one-to-one backup technique. In addition, when used in conjunction with pre-computation and pre-reservation of protection bandwidth (and bypass tunnels), facility backup can be implemented, without any or minimal routing extensions [29]. (In MPLS it is possible to pre-install a set of bypass tunnels that may share protection bandwidth, by assigning zero bandwidth [29] to each tunnel.) Moreover, by pre-reserving sufficient protection bandwidth, it can be ensured that all primary LSPs are protected, no matter how the primary path routing is done, as long as the protection capacity is not used for the primary paths. Thus, pre-reservation helps simplify network operations such as online connection routing which can now be done oblivious to the offered protection. Moreover, pre-reservation can be done by an off-line algorithm with the complete knowledge of the network. This makes it possible to maximize the bandwidth sharing among the bypass tunnels, thus minimizing the total amount of capacity that needs to be reserved for protection.

In this paper, we study the problem of determining the least amount of protection capacity (and the bypass tunnels) to be reserved in the network, so as to guarantee fast (local) reroute-based restoration for the failure of any set of $t \geq 1$ links. The solution to the problem determines for each link, its bypass tunnels and the amount of its total capacity to be reserved for protection, so that its remaining capacity can be used for carrying working traffic. Thus, in our model (as in [29]) there are (at least) two pools of bandwidth, one of which can only be used for carrying working traffic, and the other one is reserved for protection (it may carry low priority best effort working traffic that can get preempted by a rerouted flow, subsequent to a failure). The pre-reserved protection capacities of the links belong to the backup pool, and on a link failure its working traffic is rerouted, on at most k bypass tunnels, using only the available bandwidth in the backup pool. The limit of k enables the bypass tunnel information to be stored by the head nodes, of the links, in their limited memories. We show that the problem is NP-hard and we provide fast, computationally efficient algorithms, with bounded performance guarantees, for solving the problem. As shown in [29] it is not very difficult to support a distributed implementation of these algorithms since the algorithms only require the topology and the link capacities, which are available to the LSR via LSA updates. Finally, we also show how to update the pre-reserved backup bandwidth and bypass tunnels to accommodate topology changes.

Even though the results presented in this paper are in the context of MPLS networks, they are equally applicable to other technologies (e.g., Optical, ATM etc.), where local reroute may be used to provide restoration guarantees to service carrying circuits, in mesh topologies. Unlike MPLS it may not be possible to pre-install the bypass tunnels, computed by the algorithm, into the network, due to the implicit protection capacity sharing among the bypass tunnels. The pre-computed bypass tunnels can, however, be signaled at the time of failure, and the protection capacity needed for the signaled bypass tunnel, is guaranteed to be available in the backup pool.

The rest of the paper is organized as follows. Section II defines the problem and summarizes our results. In Section III, we present the background and related work. In Sections IV and V, we present efficient algorithms for the problem of single link failure and analyze their performance. In Section VI, we present algorithms for multiple link failures and analyze their performance. Section VII discusses implementation details, including how to handle changes in topology. Section VIII presents our simulation results. Section IX discusses extensions and future work. Section X concludes the paper.

II. Problem Description and Our Results

We are given a capacitated network in which pre-planned facility-based fast reroute is used to provide protection against link failures. The link capacities are assumed to be integral to model the number of fibers or the smallest switchable bandwidth on a link. The problem is to partition the link capacities into working and protection capacities (both integral) to guarantee link restoration for the failure of any set of $t \geq 1$ links, with the goal of minimizing the total amount of bandwidth used for protection. We also require that on failure of a link the working traffic of the link can be rerouted on at most k bypass tunnels. In addition, we also consider keeping the network protection capacities updated, as links are added or deleted.

We model the network as a capacitated network with no parallel links. The latter assumption is justified by the observation that in reality parallel links fail together and hence they can be replaced by a single link of total capacity equal to the sum of the capacities of the individual parallel links. Given a network with integral link capacities $u(e)$ and an integer k, the problem is to find for each link e, an integral protection capacity $0 \leq p(e) \leq u(e)$, and a set of at most k bypass tunnels $B(e)$, for protecting link e with bandwidth guarantees, such that

1) By reserving $p(e)$ of each link e's capacity for protection, the network can recover from single link failures via link-based local restoration. This means that on the failure of link e its maximum primary (working) traffic, which is $w(e) = u(e) - p(e)$, can be rerouted onto its bypass tunnels $B(e)$, and the reserved protection capacity, on the surviving links, is sufficient to meet the bandwidth requirements of the bypass tunnels. Here bandwidth sharing among bypass tunnels is assumed.
2) Each bypass tunnel of a link e, with $r(e) \leq k$ bypass tunnels, must be able to support a rerouted traffic of approximately $w(e)/r(e)$, on the failure of link e. In addition, all together these $r(e)$ bypass tunnels must be able to support the entire working traffic on link e. We will assume that on the failure of link e, an integral amount of traffic is rerouted on each of its bypass tunnels.
3) The total protection capacity $\sum_e p(e)$ is minimized.

The reason for constraining the number of bypass tunnels for each link (by the parameter k) and their minimum bandwidth, is that in practice the network may support high bandwidth connections (ATM VCs or MPLS LSPs etc.), which may not be split. Also, the head end nodes have only limited resources to store paths of too many bypass tunnels. Finally, if these tunnels cannot be pre-provisioned in the network, they have to be setup subsequent to a failure. Thus, by limiting the number of bypass tunnels per link, the desired recovery times can be achieved. Thus, ideally k should be 1. However, one advantage of having $k > 1$ is that for larger values of k, the total protection capacity

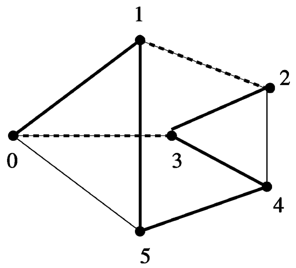

Fig. 1. A six-node network.

needed is usually much less. However, for most networks, the total protection reserved is close to the best possible, even when there are at most two bypass tunnels per link. We illustrate these observations with an example.

Consider the six-node graph given in Fig. 1, with uniform link capacities: $u(e) = 20$ units for all links e. It can be shown that when splitting is not allowed ($k = 1$), an optimal solution must set $p(e) = w(e) = 10$, for all links e, resulting in 90 units of total reserved protection capacity. Note that this is a feasible solution, since on the failure of any link, say link (1,2), its working traffic of at most 10 units can be routed on the protection capacity of a surviving path, say 1,0,3,2. Also, it can be shown (Lemma 2 in Section V-A) that when arbitrary splitting is allowed, then any solution must reserve at least 60 units of total protection capacity. This bound is also achieved with 2-splitting ($k = 2$), where $p(e) = w(e) = 10$ is set on the Hamiltonian cycle 0,1,2,3,4,5 and $p(e) = 0$ is set for all other links. In this case, on failure of any link, say (1,5), outside the Hamiltonian cycle, its working traffic, of at most 20 units, is split equally among its two incident paths, 1,0,5 and 1,2,3,4,5 on the Hamiltonian cycle. When a link, say 1,0, on the Hamiltonian cycle fails, its working traffic, of at most 10 units, is routed on the surviving Hamiltonian path, 1,2,3,4,5,0.

Thus, in practice, 2 comes close to being the best value for k, which is what we will assume in the rest of this paper. It can be shown [16] that when k is unbounded and arbitrary splitting of the rerouted traffic over the bypass tunnels is allowed (no limit on the minimum capacity of the bypass tunnels), then the above-mentioned problem can be solved optimally in polynomial time, using linear programming techniques. We show, however, that with the constraints outlined earlier, our problem is NP-complete.

One of our algorithms is applicable to networks where the splitting of the rerouted traffic is not allowed (i.e., k has to be exactly 1). This algorithm is guaranteed to produce a solution, in which there is no splitting of the rerouted traffic. However, as expected, this algorithm reserves more spare capacity for protection. Our second algorithm may create two bypass tunnels for some links in the network, but reserves close to lowest protection capacity in the network. We show that in the worst case both these algorithms produce a solution, which reserves at most twice the protection capacity of the optimal solution.

Our algorithms are very efficient to implement, which makes them amenable to devices, such as network elements (e.g., LSR), with limited computational resources.

III. Background and Related Work

In general, the protection schemes for optical and MPLS networks can be classified [14], [15] based on whether the protection is local (link based) or end-to-end (path based), and whether the backup resources are dedicated or shared. Fast or local reroute mechanisms, outlined earlier, are instances of link-based protection. In path-based protection, the entire primary service-carrying path is backed up by alternate protection paths, such that any failure on the primary path results in its traffic getting rerouted over its protection paths. In path-based protection, the reroute is done by the end nodes of the path. Compared to link-based protection, recovery may be slower in path-based protection schemes, partly because failure information has to reach the end nodes before restoration can be initiated, and partly because even a failure of a single link may affect primary paths of many different ingress–egress pairs, all of which may initiate path protection in parallel, resulting in high signaling loads and contention for common resources and crankbacks. This is the reason we focus exclusively on link restoration schemes in this work. However, path-based restoration schemes have the advantage that they are typically more efficient in terms of spare protection capacity usage. For example, Iraschko *et al.* [11] report that path protection may have as much as 19% less spare protection capacity requirement compared to link protection. Ramamurthy *et al.* [23] also report that path protection provides significant capacity savings over link protection.

The protection schemes can be further classified as being pre-planned (e.g., SONET) or event driven (dynamic). The latter involves computing bypass routes and reserving protection bandwidth at the time when the working path is provisioned. These schemes rely on heavy signaling to maintain the reservations and to effect the rerouting on the failure of a link. These schemes, although very efficient in lowering the over-build, tend to have longer restoration times. Note that our scheme is based on pre-planning with the only dynamic component coming from topology changes, due to which some protection capacities and bypass tunnels may need to be re-computed and re-provisioned.

For pre-planned facility-based fast reroute, many of the approaches are based on employing ring-like (e.g., SONET) protection mechanisms on a set of "covering" rings on the underlying mesh topology (which is assumed to be 2-edge connected). Some of these approaches only work on constrained mesh topologies that are designed in terms of rings [7], [27]. Some of these ring-based schemes are based on the notion of cycle covers, whereby rings or cycles are identified that include every link of the underlying networks [7]. Each of these cycles is then provisioned with enough protection capacity so that on the failure of any link its working traffic can be rerouted over the protection capacities in the surviving links of its covering cycles. There are two drawbacks of this problem: one, the over-build can be significant, and two, it is NP-hard to find the smallest cycle cover of a given network [28]. An improvement to cycle cover called the p-cycle [8] is based on the observation that a cycle can be used to protect not just the links of the cycle but also any other link whose end nodes are contained in the cycle, thus suggesting that far fewer cycles may be sufficient for providing full protection. An algorithm to minimize the total spare capacity, based on solving an integer program over all possible cycles, is given in [8]. To the best of our knowledge, no fast approximation algorithms are known for this problem. Another alternative approach similar to cycle cover, called the double cycle cover, finds a set of cycles that include each link in exactly two cycles [6]. A double cycle cover can be found in polynomial time for planar graphs and it is conjectured that double cycle covers exist for all 2-edge connected graphs [12], [26]. This is in contrast to the cycle cover problem which is known to be NP-hard. However, even for double cycle cover-based protection schemes, the required network over-build can be significant. Note that all the ring-based approaches suffer from the drawback that after any topology change, the structure of the solution may change dramatically, thus limiting their scalability.

Non-ring-based approaches to link restoration on mesh networks include generalized loop-back [19], [20], where an orientation of the edges is selected to form a digraph, called the primary. A conjugate digraph called the secondary is then obtained by orienting the edges of the graph in the opposite direction. This edge orientation is chosen to ensure that the links on the secondary can be used to carry rerouted traffic for any link failure in the primary. Reference [2] considers the problem of finding the minimum cost augmentation of a given primary network, so that the resulting network is capable of supporting link protection under single link failures, for a given set of links. In their model, no limit is imposed on the capacities of the links, and they provide a 4-approximation algorithm when all links in the primary network have uniform bandwidth and they provide a 10.87-approximation algorithm for the general case. In addition, [2] also provides a $O(\log n)$-approximation algorithm for the problem of jointly designing the primary and backup networks.

All the schemes mentioned earlier assume that protection is provided for a single link failure. The work of [17] evaluates the robustness of link restoration schemes in the presence of multiple link failures. A hierarchical classification of the reasons due to which restoration algorithms fail to guarantee recovery for multiple link failures is provided and is illustrated for some standard networks. In [23], the authors find that path protection is more susceptible to multiple link failures than link restoration and dedicated protection is more resilient than shared protection to multiple link failures. In [30], the authors propose a scheme to deal with multiple link failures by re-provisioning protection paths for connections that become vulnerable or are left unprotected from subsequent failures. The work of [18] explores the tradeoff between spare capacity for protection and the robustness to double link failures. A restoration scheme is presented that extends generalized loop-back to operate on a subgraph of the full backup graph, thus providing savings on the spare capacity reserved for protection. These results indicate that the modified scheme has equivalent or better robustness to double link failures as the original generalized loop-back scheme and can provide an additional 20% capacity to carry unprotected traffic. The ability of p-cycles to survive double link failures is considered in [24]. Note that the existing work mentioned so far mainly considers the robustness to multiple link failures of protection schemes designed for single link failures. This is in contrast to our protection scheme and the one presented in [3], which guarantee 100% or exhibit almost 100% robustness, respectively, to multiple link failures.

Recently, [16] also designed pre-planning schemes for supporting fast reroute. This model and problem setting is very close to ours except for one major difference. In our scheme, we can restrict the amount of splitting that must be incurred by the rerouted traffic, while in their case no such restriction may be imposed. In other words, in their scheme no bound may be imposed on the number of bypass tunnels needed to reroute the traffic after a failure. This difference has a big impact on the complexity of the problem since for the model considered by [16] the problem can be solved in polynomial time by using a linear programming approach. However, the problem is NP-hard in our setting and hence very different solution techniques are needed.

Putting our results in perspective of the existing schemes described earlier, our algorithms may reserve only a fraction of the total capacity of the network for protection. On the other hand, all the ring-based schemes (with the exception of p-cycle) and those based on generalized loop-back, may reserve at least half the total capacity for protection. For example, for a network with uniform capacities, n nodes and m links, our algorithms may reserve at most n capacity for protection, while most ring-based schemes (including SONET) will reserve $m/2$ capacity for protection. Note that m can be arbitrarily large compared to n, depending on the average degree of the network. We also show that changes in topology can be easily handled with our solution, which is not always the case for the existing schemes.

IV. Algorithms for Single Link Failure

In this section, we present two fast algorithms, for the problem of minimizing the total amount of pre-provisioned protection capacity, and for computing the set of pre-installed bypass tunnels, to ensure that the network is fully link protected. We establish that both the algorithms have the same worst case performance. However, the two algorithms obtain quite different solutions, where one algorithm reserves either all or none of the capacity of every link for protection, while the other one ensures that only a portion of any links capacity is reserved for protection. Also, one algorithm only outputs a single bypass tunnel per link, while the other algorithm may require that on failure of some set of links, the traffic is rerouted over two bypass tunnels, resulting in much lower total reserved protection capacity on the links. Thus, depending on the needs of the service provider, one algorithm may be better suited than the other. We show that the solutions output by both the algorithms reserve no more than twice the protection capacity reserved by any optimal solution.

Let the given undirected network be denoted by $G = (V, E)$, where V is the set of vertices and E the set of bidirectional links. Recall that $u(e)$ denotes the total capacity of link e. We use the notation $p(e)$ and $w(e)$ to denote the protection and working capacities on link e, as assigned by the algorithm. Note that $u(e) = p(e) + w(e)$. Let $e_1, e_2, \ldots, e_m$ denote an ordering of the links in non-increasing order of their capacities. Thus, for $i > j$, we have $u(e_i) \leq u(e_j)$.

A. Algorithm Based on Spanning Tree Construction

We assume without loss of generality that G is connected. This is because otherwise the algorithm can be independently run on each connected component. The algorithm maintains an acyclic graph (collection of forests) T, where T initially consists of only the nodes V, and on termination is a spanning tree of G. At step i, link e_i is considered, and if it does not create a cycle in T then it is added to T. Thus, after m steps, all links are considered and T is a tree. The algorithm then sets $p(e) = u(e)$, for all links in the tree and sets $p(e) = 0$, for all other links. Note that since there is never any working traffic carried on the links of T, there is no need to provide any protection for a failure of any such link. Thus, for these links, there are no bypass tunnels. For a link $e = (u, v)$, which is not in T, its single bypass tunnel is the unique path from u to v in T. Algorithm 1 describes the algorithm.

Algorithm 1 Algorithm TREE

```
Let $\{e_1, e_2, \ldots, e_m\}$ be the links sorted in non-increasing
order of capacities.
$T = \phi$
for $i = 1, \ldots, m$
{
    if $(T \cup \{e_i\})$ does not form a cycle
    {
        $T = T \cup \{e_i\}$
        $w(e_i) = 0$
        $p(e_i) = u(e_i)$
    } else {
        $w(e_i) = u(e_i)$
        $p(e_i) = 0$
    }
}
```

Validity of the Algorithm: We show that the working traffic on all links in the solution output by the algorithm is link protected, thus establishing that the solution is feasible. In other words, we show that if a link e with working traffic $w(e)$ is cut, then its bypass tunnel is able to support a flow of $w(e)$ unit. Thus, if e' is a link on the bypass tunnel for link e, then we have to show that $p(e') \geq w(e)$. However, since the links on the bypass tunnel are links in T, they have $p(e') = u(e')$. Also, by construction, the links e that need protection are not in T, and they have $w(e) = u(e)$. Hence, we have to show that $p(e') = u(e') \geq w(e) = u(e)$. We prove this by contradiction. So, let $u(e') < u(e)$. Let $e = (u, v)$. Note that e' is a link in the unique path from u to v in T. Since $u(e') < u(e)$, the algorithm must consider link e before link e'. Since link e is not added to T, there must exist a path connecting node u and v in T, when link e is considered, and also when link e' is considered. But then, adding e' to T would have created a cycle. Hence, e' cannot be in T, which is a contradiction. We show later that the solution output by this algorithm uses no more than twice the optimal protection capacity.

Time complexity: The algorithm finds a maximal spanning tree, similar to Kruskal's algorithm. Hence, the time complexity is $O(m \log n)$.

Enhancements: Note that the algorithm sets the working capacity of the tree links to zero. If the network is very sparse, we can assign the working capacities for tree links as follows. Assign a working capacity of ϵ to all the tree links and a protection capacity of ϵ to the remaining links. It is easy to see that as long as $m \leq 2(n-1)$, the total protection capacity will not increase, and as long as $\epsilon \leq 1/2\times$ capacity of the lowest capacity link, the solution is a feasible solution with $k = 1$. For example, this enhancement will assign half protection capacity on all the links for a ring topology.

B. Algorithm Based on 2-Edge Connected Graph Construction

We assume without loss of generality that G is 2-edge connected.

Algorithm 2: Algorithm 2-EDGE

```
Let $\{e_1, e_2, \ldots, e_m\}$ be the links sorted in non-increasing
order of capacities.
$F$ = Tree $T$ output by 1
$M = \phi$ /* Links to which protection capacities are assigned */
for $i = 1, \ldots, m$ such that $e_i \notin T$ {
    Let $e_i = (v_1, v_2)$
    if ($v_1$ and $v_2$ are not 2-edge connected in $F$) {
        $F = F \cup \{e_i\}$
        $C$ = Unique cycle formed by adding $e_i$ to $F$
        for each link $e \in C - M$ {
            $w(e) = u(e_i)/2$
            $p(e) = u(e) - w(e)$
            $M = M \cup \{e\}$
        }
    } else {
        $w(e_i) = u(e_i)$
        $p(e_i) = 0$
    }
}
```

Algorithm 2 describes the algorithm. It starts from the tree T, created by the first algorithm, and adds more links to it, as follows. At all times, the algorithm maintains a connected graph F. F is initially set to T. The algorithm considers the links e_i not in T, in the order of increasing index i, and hence in the order of non-increasing capacity. If, while considering link $e_i = (v_1, v_2)$, the nodes v_1 and v_2 are not 2-edge connected in F, then link e_i is added to F. We say a pair of vertices v_1 and v_2 are 2-edge connected in F if removal of any single link in F does not disconnect v_1 from v_2.

For computing the backup capacities of the links in F, we keep track of the set of links that has been assigned backup capacities in a set M. Initially, M is empty. When a link e_i is added to F, there is a unique cycle C in F which contains link e_i; all other links in C are from T. For all the links e in C that are not assigned any protection capacities (i.e., links in the set $C - M$), we allocate a working capacity of $w(e) = u(e_i)/2$ and a protection capacity of $p(e) = u(e) - w(e)$. We also add e to M, the list of links that are assigned a protection capacity. The backup tunnel for e is the path formed in C when e is deleted from C.

The algorithm sets the protection capacity of all the links that are not in F to zero. Each of these links e is assigned two bypass tunnels, as follows. Let $e = (u, v)$. Note that when e is considered by the algorithm (while constructing F), nodes u and v are 2-edge connected in F (that is why e is not in F). Thus, when e is considered by the algorithm, there must exist two link disjoint paths between u and v in F. The two bypass tunnels for link e are these two paths.

Time Complexity: The algorithm iterates through all the links $e_i = (v_1, v_2)$ and checks whether the end points v_1 and v_2 are 2-edge connected in F. It also computes the path between the end points in F. These two operations can be done in $O(m)$ time as follows. The path P between v_1 and v_2 can be computed in $O(m)$ time using a depth first search (DFS) of F from one of v_1 or v_2. It can be shown that v_1 and v_2 are 2-edge connected in F only if the end points of all the links in P are 2-edge connected. The end points of a link in F are 2-edge connected only if the link is in M. Hence, the time complexity of the algorithm is $O(m^2)$.

Example: We illustrate the algorithm using Fig. 2. Here, each link has two labels—the first one is its total capacity and the second one is its protection capacity, as set by the algorithm. The maximum cost spanning tree T, as found by the algorithm, is shown in thick solid lines. F is initially set to T. Next, the algorithm considers the remaining links in the decreasing order of their capacities (i.e., links (1,2), (2,4), (1,5) and (3,4)). When link (1,2) is considered, its end points are not 2-edge connected in F. Hence, it is added to F and the cycle C formed is 1-0-3-2-1. Since none of these links are in M, the working capacity of each of these links is set to half of capacity of link $(1, 2)$, which is 4. They are all added to M. Link (2,4) is the next link considered and since its end points are not 2-edge connected, it is added to F. 2-3-0-5-4-2 is the unique cycle C in F containing (2,4). The links of C, which are not assigned protection capacity (i.e., not part of M) are (2,4), (0,5), and (4,5). These are assigned a working capacity 4, which is half of capacity of link (2,4). Links (1,5) and (3,4) are not added to F, since their end points are 2-edge connected in F, when they are considered. They are given full working capacity.

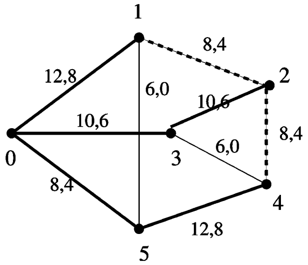

Fig. 2. A six-node network showing the capacities.

Validity of the Algorithm: We first show that the graph F output by the algorithm is 2-edge connected. Note that if there is a cut link (u, v) in F, then (u, v) must also be in T (since u and v must be connected in T). Since G is assumed to be 2-edge connected, there must be some link e_j in G that is not in T or F, which when added to T must create a cycle containing link (u, v). Let e_j be such a link, with the smallest index j. Then, link e_j must have been considered by this algorithm, and at the time when it is considered by the algorithm, u and v cannot be 2-edge connected in F. Hence, the algorithm must add e_j to F. But then (u, v) cannot be a cut link of F, a contradiction. Now we show that in F all links are protected.

Lemma 1: The algorithm outputs a feasible solution, in which the working capacity of all links is protected.

Proof: We first show that on any link in F, at least half of its capacity is reserved for protection. This holds trivially for the links in F that are not in T. For a link e in T, its protection capacity is assigned by the algorithm while considering some link e_i, and cycle C, such that e is in $C(e)$. Note that since e is in T, we have $u(e) \geq u(e_i)$. By construction link e's protection capacity is set to $u(e) - u(e_i)/2 \geq u(e)/2$, thus implying the result.

Next, we show that working capacities of all the links in F are protected. The links e in F are protected by the path in the first cycle C formed in F that includes e. The links $e = (v_1, v_2)$ that are not part of F are protected by two paths present in F between v_1 and v_2 when e is considered by the algorithm. It is easy to see that these paths have at least $u(e)/2$ protection capacities, since e is considered after all the links in the two paths. ■

C. Uniform Capacity Case: Practical Consideration

In the case when all the links have the same total capacity ($u(e) = u$, for some integer u), then both the algorithms may consider the links in any arbitrary order. The worst case guarantees, which we show later, hold for any such order. However, in practice some orderings may be better than other ones. Here we present a scheme based on one such ordering. Note that when all links have the same capacity u, the algorithm sets $p(e) = u/2$ for all links in F, and $p(e) = 0$ for all links outside F. Thus, the amount of protection capacity reserved by the algorithm is directly proportional to the number of links in the 2-edge connected graph F.

In order to minimize the number of links in the 2-edge connected graph F, we propose the following algorithm. Let T be obtained by doing a DFS on G. The algorithm to construct F, starting from T, is modified as follows. At any step, the algorithm considers that link $e=(u,v)$ to add to F, for which u and v are not 2-edge connected in F, and the number of links on the unique path from u to v in T that are not in any cycle of F is maximized. We can show that the worst case performance of this heuristic is the same as the algorithm outlined before. However, in practice this algorithm finds solutions with lower total protection capacity. As an example, consider the network in Fig. 1, with uniform link capacities. The DFS tree T is shown in thick solid lines. The algorithm adds links (0,3) and (1,2) (shown as thick dashed lines) to T, in that order, to construct F. All these links in F have half their capacity reserved for protection, and the remaining two links in G have no capacity reserved for protection. Thus, the total protection capacity reserved by the algorithm is at most 16.7% more than the amount reserved by the optimal solution.

V. Analysis

In this section, we show that all the algorithms presented in Section IV have good worst case performance. Specifically, we show that these algorithms are guaranteed to find a solution with total protection capacity no more than twice that of the optimal solution.

First, we establish a lower bound on the amount of total protection capacity that is needed by any solution.

A. Lower Bound

Let the network have n nodes denoted by the set V. Let $\delta(v)$ denote the set of links incident on node v. Let the maximum capacity of any link incident on node v be $M(v)$. Thus, $M(v)=\max_{e\in\delta(v)} u(e)$.

Lemma 2: Any solution must reserve at least $\sum_{v\in V} M(v)/2$ total protection capacity on the links of the network.

Proof: The following proof applies, even when there is no limit on the number of bypass tunnels for the links, and even when the working traffic is split arbitrarily among the bypass tunnels. Consider any solution. Let v be a node and let e be a link of capacity $u(e)=M(v)$, incident on node v. On the failure of link e, the working traffic on link e must be rerouted over the remaining links in $\delta(v)$. Since the working traffic of link e can be as large as $u(e)-p(e)$, the sum of the protection capacities of the remaining links in $\delta(v)$ must be at least $u(e)-p(e)$. Thus, the total protection capacity on all the links in $\delta(v)$ must be at least $u(e)-p(e)+p(e)=u(e)=M(v)$. Consider the sum $2\sum_e p(e)$. Note that this equals $\sum_{v\in V}\sum_{e\in\delta(v)} p(e)\geq\sum_{v\in V} M(v)$. Thus, $\sum_e p(e)\geq\sum_{v\in V} M(v)/2$. ■

Corollary 3: When all links have the same capacity $u(e)=u$, for all e, then at least $un/2$ total protection capacity is reserved by any solution.

Now we show that the algorithms described earlier are 2-approximation algorithms.

B. Algorithm Based on Spanning Tree Construction

Let T be the tree (forest) found by the algorithm. We first show that the total capacity of the links of T is at most $\sum_{v\in V} M(v)$.

Lemma 4: The total capacity of the links in T is at most $\sum_{v\in V} M(v)$.

Proof: The proof uses a charging argument, where the capacity of each link in T is charged to at least one vertex in V, such that the total capacity charged to each vertex v is at most $M(v)$. This implies that the total capacity of the links of T is at most $\sum_{v\in V} M(v)$. The charging works as follows. Let u be some arbitrary vertex in V. Let S be a subset of vertices, which is initially set to $S=\{u\}$, and in the end is equal to V. At each step the charging scheme picks one unpicked link of T that connects some vertex in S to some vertex not in S. Note that such a link must always exist as long as there is at least one unpicked link of T. Let the charging scheme pick link $e=(x,y)$ with x in S and y not in S. The capacity of link e is charged to vertex y. Thus, y gets a charge of $u(y)$, which is at most $M(y)$, since link e is one of the links in $\delta(y)$. At this point S is set to $S\cup\{y\}$, and the charging scheme continues by picking another unpicked link from T that connects some vertex of S to some vertex not in S. Note that in this charging scheme, each vertex is charged at most once, since it is charged only when it is brought into S. Also, as shown above, the charge on any vertex y is at most $M(y)$, thus establishing the result. ■

Theorem 5: The spanning-tree-based algorithm is a 2-approximation algorithm.

Proof: Follows from Lemmas 2 and 4. ■

C. Algorithm Based on 2-Edge Connected Graph Construction

Theorem 6: The algorithm based on 2-edge connected graph construction is a 2-approximation algorithm,

Proof: Recall that this algorithm starts out with the tree T, created by the first algorithm, and adds more links to it, while adjusting the reserved protection capacity on the tree links and the newly added links. As shown in Theorem 5, the total protection capacity reserved (which is all on T) by the spanning-tree-based algorithm is at most twice the protection capacity reserved by an optimal solution. By using a charging argument, we show that as link protection capacities are updated by this algorithm, the total protection capacity does not increase, thus implying that the total protection capacity of the solution output by this algorithm is also at most twice the protection capacity reserved by an optimal solution.

The algorithm can be thought of as starting with full protection for T, and when a link which is not part of T is added to F, some of the protection capacity from T is transferred to that link. When link $e_i=(v_1,v_2)$ is added to F, then the protection capacity of links in C–M is decreased by $u(e_i)/2$ and the link e_i is assigned a protection capacity of $u(e_i)/2$. We claim that there is at least one link e in T that is in C–M. The proof is by contradiction. If there is no such link e, then just before e_i is added by the algorithm to F, each of the link on the path joining v_1 with v_2 in T is already in some cycle. Thus, v_1 and v_2 are 2-edge connected just before link e_i is added by the algorithm. However, in this case, the algorithm would not add link e_i to F, a contradiction. ■

It can be shown that the uniform capacity case given in Section IV-C is also a 2-approximation algorithm.

D. NP-Completeness Result

In this section, we show that even a simple version of the problem is NP-complete.

Claim 7: For a given value P, the problem of determining if there exists a solution that reserves at most P total protection is NP-complete for $k = 2$. Furthermore, this holds even when all $u(e)$ are equal.

Proof: Note that given a solution to the problem (the protection capacities on each link and the bypass tunnels for each link), it can be verified in polynomial time if it is a feasible solution for protecting against any single link failure, and hence the problem is in NP.

Consider an instance of the problem for $k = 2$ with all edge capacities $u(e) = 2$. In any solution to this problem, each link's working traffic is rerouted (split equally into integral flows) on at most two bypass tunnels.

We reduce the problem of determining if there exists a Hamiltonian circuit in a given connected graph to this problem. The reduction sets $P = n$ and sets every link capacity to 2. We claim that the given connected graph has a Hamiltonian circuit if and only if the reduced instance has a solution of total protection capacity at most n. Let the graph have a Hamiltonian circuit. We set $p(e) = w(e) = 1$ for all links e in the Hamiltonian circuit, and we set $p(e) = 0$ and $w(e) = 2$ for all the other links e. Note that this solution has total protection capacity exactly $n = P$. Each link e in the Hamiltonian circuit has a single bypass tunnel, which is the Hamiltonian path obtained by removing e from the Hamiltonian circuit.

A link (u, v) which is not on the Hamiltonian circuit has two bypass tunnels, corresponding to the two paths connecting node u to v in the Hamiltonian circuit. Thus, the bypass tunnels only use the links of the Hamiltonian circuit, each of which has one unit of capacity reserved for protection. It is easy to see that this is a feasible solution.

The proof in the other direction works as follows. Let the optimal solution of the reduced instance reserve at most P total protection capacity. Note that by Corollary 3, any solution to this instance must use at least $n = P$ protection capacity. Hence, the optimal solution must use exactly $P = n$ protection capacity. Consider any link e with $p(e) = 2$ in the optimal solution. Let $e = (u, v)$. Then it must be the case that no other link e' incident on node u or node v can have $p(e') > 0$ in this solution. This is because, as shown in the proof of Lemma 2, for every node w we have $P(w) \geq M(w) = 2$, where $P(w)$ is the total protection capacity on the links incident on node w (links in $\delta(w)$). Thus, if some link e', other than link (u, v), incident on say node u has $p(e') > 0$, then $P(u) > 2$. In that case, the total protection capacity reserved by the solution, which is shown in the proof of Lemma 2 as at least $\sum_v P(v)/2$, would be strictly greater than n, leading to a contradiction. Thus, neither node u or node v can have another link e incident on it, with $p(e) > 0$. A consequence of this is that no bypass tunnel, in the optimal solution, can contain a link e for which $p(e) = 2$. This is because a bypass tunnel must have at least two links, each with strictly positive protection capacity reserved on it. Thus, a link e with $p(e) = 2$ is not useful to any solution, implying that by setting $p(e) = 0$, we can decrease the cost of the optimal solution, while not changing its feasibility. Thus, there must not exist any links e with $p(e) = 2$ in the optimal solution.

A similar argument shows that, in the optimal solution, for any node v there are at most two links e in $\delta(v)$ with $p(e) = 1$. Let S be the set of links with $p(e) = 1$ in the optimal solution. Since there are no links with $p(e) = 2$ in the optimal solution, we must have that the number of links in S is exactly n. Note that the graph formed by the links in S is connected, since all the bypass tunnels must only use links in S. Therefore, if it has two or more connected components, say S_1 and S_2, then since the original graph is connected, there must exist a link e which is not in S, with one endpoint in S_1 and the other endpoint in S_2 with $w(e) = 2$. Note that link e is not protected in the optimal solution, and hence such an e does not exist. Moreover, the graph formed by links in S is 2-edge connected. This is because, otherwise the working traffic (of one unit) on the failure of a cut link e of S cannot be routed over the protection capacities on the surviving links (remaining links in S).

The only possible solution with these properties for S (n links and 2-edge connected) is that the links of S must form a Hamiltonian circuit. Thus, the given graph must have a Hamiltonian circuit. ■

VI. Algorithms for Multiple Link Failures

There has been some work on survivability of networks against multiple link failures [3], [4], [18], [25], [30]. So far, we have presented our results for the basic version of the problem, where we want to protect against a single link failure. However, our results also extend to the case where we want to protect against multiple link failures. We now show how to extend our algorithms and their analysis to deal with $t > 1$ link failures. We start with the algorithm based on the spanning tree construction (Section IV-A). Let $e_1, e_2, \ldots, e_m$ denote an ordering of the links in non-increasing order of their capacities. Thus, for $i > j$, we have $u(e_i) \leq u(e_j)$. The new algorithm works very much the same way as the algorithm for the single link failure except that it computes multiple (t) different acyclic graphs (forests) $T_1, T_2, \ldots, T_t$, instead of just a single acyclic graph T, such that no pair of graphs T_i and T_j share any links. Initially, each of the acyclic graphs consists of only the nodes of V and has no links. At step i, link e_i is considered for the graphs $T_1, T_2, \ldots, T_t$ in that order. Link e_i is added to the first graph T_j (if one exists) in which link e_i can be added without creating a cycle. After m steps the algorithm sets $p(e) = u(e)$ for all links in the graphs $T_1, T_2, \ldots, T_t$ and sets $p(e) = 0$ for all other links. Algorithm 3 describes the algorithm.

Algorithm 3: Algorithm MULTI-TREE

```
Let ${e_1, e_2, \ldots, e_m}$ be the links sorted in decreasing order
of capacities.
Let $T_i = \phi$ for all $i = 1, \ldots, t$
for $i = 1, \ldots, m$ {
    for $j = 1, \ldots, t$ {
        if ($T_j \cup \{e_i\}$ does not form a cycle) {
            $T_j = T_j \cup \{e_i\}$
            $w(e_i) = 0$
            $p(e_i) = u(e_i)$
            break;
        }
    }
    if ($e_i$ was not added to any of $T_j$) {
        $w(e_i) = u(e_i)$
        $p(e_i) = 0$
    }
}
```

We first show that the algorithm finds a feasible solution. Consider a failure of t links $e_{i_1}, e_{i_2}, \ldots, e_{i_t}$. Without loss of generality assume that none of the links $e_{i_1}, e_{i_2}, \ldots, e_{i_r}$ is in any of the acyclic graphs $T_1, T_2, \ldots, T_t$ and each of the remaining links $e_{i_{r+1}}, \ldots, e_{i_t}$ is in one of $T_1, T_2, \ldots, T_t$. Note that there must be at least r acyclic graphs such that the set of links $e_{i_{r+1}}, \ldots, e_{i_t}$ are not in any of these graphs. Again without loss of generality let these graphs be $T_1, T_2, \ldots, T_s$ for some $s \geq r$. Consider link $e_{i_j} = (u, v)$ for $j \leq r$. Note that there must be a path P_j joining the end nodes u and v in T_j, because otherwise while considering link e_{i_j} the algorithm must have added it to the graph T_j. By construction all the links on this path P_j have all of their capacity reserved for protection and this capacity is at least $u(e_{i_j})$. Since T_j does not have any failed link, there are no failed links on path P_j. In other words, none of the links in $e_{i_1}, e_{i_2}, \ldots, e_{i_t}$ are on P_j. Hence, path P_j can serve as a bypass tunnel for link e_{i_j}. Thus, edge disjoint bypass tunnels for all links $e_{i_1}, e_{i_2}, \ldots, e_{i_r}$ can be found. Note that since all links in $e_{i_{r+1}}, \ldots, e_{i_t}$ have zero working capacity, the working traffic on these links is zero, and hence no bypass tunnels are needed for them. Thus, all the working traffic effected by the failure can be safely rerouted.

In general, for any link e_i that is not in any of $T_1, T_2, \ldots, T_t$, one can find t edge disjoint bypass tunnels B_i, such that tunnel $b_i^j \in B_i$ consists of only edges from graph T_j. In addition, for the failure of any t links $e_{i_1}, e_{i_2}, \ldots, e_{i_r}, \ldots, e_{i_t}$ where the links $e_{i_1}, e_{i_2}, \ldots, e_{i_r}$ are not in any of $T_1, T_2, \ldots, T_t$ and the remaining links are in some $T_1, T_2, \ldots, T_t$, there exists one bypass tunnel per e_i from the set B_i such that these bypass tunnels are edge disjoint and they can carry the working traffic on the corresponding failed link e_i.

Time Complexity: The algorithm maintains t disjoint forests and its running time is equivalent to running t instances of Kruskal's algorithm. Hence, the time complexity is $O(tm \log n)$.

We now show that the extended algorithm is also a 2-approximation algorithm for protecting against any $t > 1$ failures. We first extend the lower bound in Section V-A (Lemma 2). As before, let $\delta(v)$ denote the set of links incident on node v. Let $d(v)$ denote the degree of node v. Let $\delta^s(v)$ denote a set of s largest capacity links in $\delta(v)$. If $s > d(v)$, $\delta^s(v)$ contains all links of $\delta(v)$. Let $M(v)$ be a node capacity function defined as $M(v) = \sum_{e \in \delta^t(v)} u(e)$: the total capacity of all links in the set $\delta^t(v)$.

Lemma 8: Any solution must reserve at least $\sum_{v \in V} M(v)/2$ total protection capacity on the links of the network.

Proof: Consider a node v. Let $d(v) \geq t + 1$. Consider the t links in $\delta^t(v)$. On their failure, the working traffic on these t links must be carried over the protection capacity on the remaining $d(v) - t$ links. Let W and P be the total working and protection capacity on these t links, respectively. Note that $W + P = M(v)$. Thus, the total protection capacity on the remaining $d(v) - t$ links is at least W. Hence, the total protection capacity on the links incident on node v is at least $W + P = M(v)$. Now let $d(v) \leq t$. None of these links must carry any working traffic since all these links may fail together and on failure of all these links the working traffic on these links cannot be rerouted. Hence, the total protection capacity on the links incident on node v is at least $M(v)$. Adding this for all nodes and noting that each link's protection capacity is counted twice in the sum, we get that the total protection capacity on the links of the network must be at least $\sum_{v \in V} M(v)/2$ to protect against t link failure. ■

Next, we show that the total capacity of the links in the graphs $T_1, T_2, \ldots, T_t$ is at most $\sum_{v \in V} M(v)$. Note that since only these links have capacity reserved for protection by the algorithm, this would imply that the algorithm reserves no more than $\sum_{v \in V} M(v)$ total protection capacity, which is at most twice the total protection capacity reserved by any optimal solution. We denote the set of links in graph T_i by E_i.

Claim 9: The total capacity of the links in $\cup_i E_i$ is at most $\sum_{v \in V} M(v)$.

Proof: The proof uses a charging argument where the capacity of each link in $\cup_i E_i$ is charged to at least one vertex such that the total capacity charged to each vertex v is at most $M(v)$. This thus shows that the total capacity of all links in $\cup_i E_i$ is at most $\sum_{v \in V} M(v)$

Consider acyclic graph T_i. We consider a connected component C_i^j of T_i. Note that C_i^j forms a tree. Let E_i^j be the set of links of E_i restricted to this connected component. Let $V^j \subseteq V$ be the set of nodes spanned by the tree C_i^j. We now invoke the charging scheme in the proof of Lemma 4 where it is shown that the capacity of the links in E_i^j can be charged to the vertices in V^j such that each vertex in E_i^j is charged at most once for the capacity of a distinct link in E_i^j. In other words, it is shown that the charging guarantees that the capacity of each link (u, v) in E_i^j is charged to exactly one vertex on which it is incident (u or v) and each vertex is charged at most once for one link in E_i^j.

This charging is repeated for each connected component of T_i. Since the connected components are disjoint, the charging guarantees that the capacity of each link (u, v) in E_i is charged to exactly one vertex on which it is incident (u or v) and each vertex in V is charged at most once for one of its incident links in E_i.

The charging is repeated for each T_i. Thus, overall the charging guarantees that the capacity of each link (u,v) in $\cup_i E_i$ is charged to at least one vertex in V and each vertex in V is charged for the capacities of at most t distinct link in $\cup_i E_i$ which are incident on it. However, note that $M(v)$ is at least the total capacity of any (at most) t edges incident on v. Thus, the total charge on any node v is at most $M(v)$. Since the capacity of every link in $\cup_i E_i$ is charged to some node $v \in V$, we have the total capacity of all links in $\cup_i E_i$ is at most $\sum_{v \in V} M(v)$. ■

Remark: It is possible that the algorithm described above may yield a degenerate solution in which all edges are assigned zero working capacity (if the t acyclic graphs $T_1, T_2, \ldots, T_t$ include all the edges). This may happen for instance if the given graph lacks sufficient connectivity for a non-degenerate solution to exist (for example, if the given graph has no cycles, then no working traffic can be protected even for $t = 1$). However, this may also happen even for $t + 1$ connected graphs. Although, for such graphs it must be the case that in any feasible solution at least half of the total edge capacities must be dedicated for protection (since the algorithm is a 2-approximation). For such graphs, alternative non-degenerative solutions may be found by other means. For example, for such graphs with uniform edge capacity a feasible (and optimal) solution is to set each edge's working capacity to half its total capacity. Such alternative solutions are left for future study.

The algorithm based on 2-edge connected construction (Section IV-B) naturally extends to deal with $t > 1$ link failures. Since the extension mirrors the extension for the algorithm based on the spanning tree construction, we omit the details. Similarly, the proof for the 2-approximation ratio for the extended algorithm is along the line of the proof given in Section V-C and is thus omitted.

VII. Implementation Issues

So far, we have looked at the problem of computing the initial set of protection capacities and bypass tunnels, to be pre-provisioned in a network at startup. These initial set of values may be pre-provisioned in the network by a management system. Subsequent updates to these, to deal with changes in topology, may be performed by the network in conjunction with the management system. To this end, the reserved link protection capacities may be advertised as part of the LSA. In order to ensure there is no over-subscription of protection bandwidth, the bypass tunnels may be pre-provisioned, in the network, with zero bandwidth each [29].

Next, we consider topology changes and describe our algorithm to deal with them. The algorithm computes updated protection capacities and bypass tunnels following a topology change. We assume a central server model for computing the updated values, where the server is implemented as a Label Switch Router (LSR) in the MPLS network [29]. The algorithm can be modified to operate in a distributed implementation. However, for ease of exposition we will assume a centralized model.

The LSR server monitors the LSA updates from the network to identify changes in topology. Following a topology change, it re-computes the new solution and updates the network with the new solution. For ease of presentation, we only describe the high level ideas, for two basic topology update operations: the addition of a link and the deletion of a link. In the following, we will assume $k = 2$ and single link failure. The case when at most one bypass tunnel is allowed per link and multiple link failures can be similarly handled.

When a new link $e = (u,v)$ is added, the only update to the solution is the protection capacity and the bypass tunnel for link e. The amount of protection capacity reserved on e depends on how much protection capacity is currently available between nodes u and v in the network. This can be determined using a max-flow computation on the protection capacities of the links of the network. Depending on how much protection capacity is available, the algorithm computes a lower bound on the amount of protection capacity to be reserved on the link. Having determined a lower bound on the protection capacity for link e, and hence an upper bound on the working capacity of link e, the algorithm attempts to maximize the amount of working capacity (up to the upper bound) that can be assigned to link e without changing any other links protection or working capacity. To test whether a given amount w of working capacity can be assigned to the link, the algorithm solves a max-flow problem on an auxiliary unit capacity graph. In addition to determining if w is feasible, the max-flow computation also yields the (at most two) bypass tunnels for link e when w is used as working capacity on link e. Next, to maximize the working capacity w that can be assigned to link e (up to the upper bound), the algorithm uses a binary search on the range of allowed values for w.

Next, we consider the case when a link e is deleted from the network. The easy case is when none of the bypass tunnels of any of the surviving links contains link e. In this case, there is nothing to be done. However, note that even if a bypass tunnel of link e' contains link e, link e' may still be protected in the new network, since there may exist another bypass tunnel for link e' using only the protection capacities of the surviving links. In this case, only the bypass tunnels for link e' need to be updated. This can be done easily by considering e' to be a newly added link to the new network and then by using the procedure described earlier for the link addition case to determine whether link e' can be assigned a working capacity of $w(e')$. Note that this way, the new bypass tunnels for link e' can also be computed. Now consider the case when there is at least one link e' that is not protected in the new network. In this case, the protection capacities of other links have to be updated as well. Note that the amount of slack on a link, for the protection capacity, is the difference of its working capacity and the amount of working traffic currently being carried by the link. This gives an upper bound on how much the link's protection capacity can be increased. The algorithm for updating the links protection capacity operates in two phases. In the first phase, all the link protection capacities are uniformly increased (including link e') to their upper bound until all links e' are protected. In the second phase, the algorithm lowers the protection capacity of those links that have a slack in their protection capacity. Finally, the algorithm updates the bypass tunnels for all the links.

VIII. Simulation Results

To measure the performance of our algorithms, we did extensive simulations using various real and simulated networks.

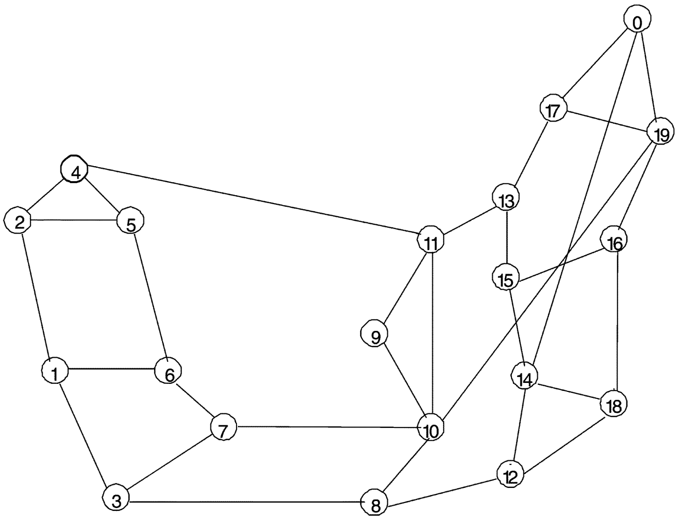

Fig. 3. ARPANET network.

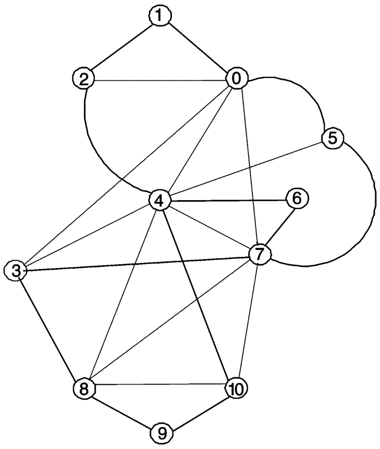

Fig. 4. NJ LATA network.

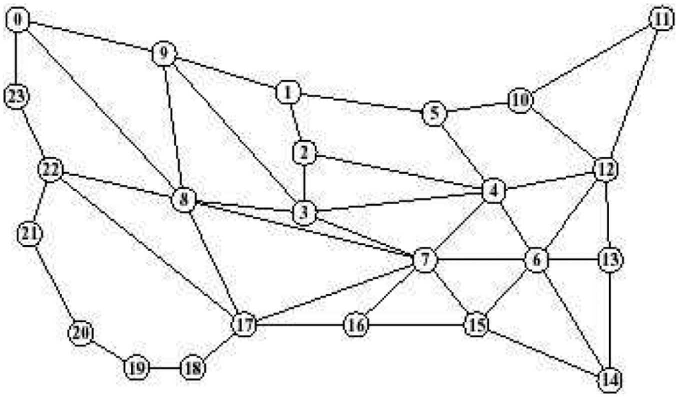

Fig. 5. National network.

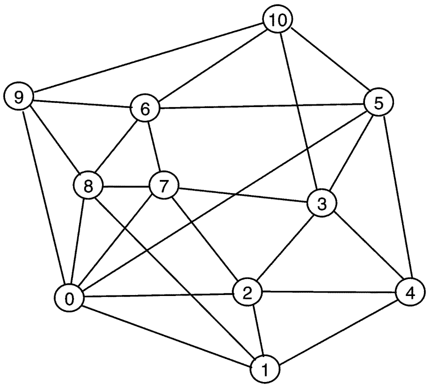

Fig. 6. European Cost239 network.

Here we only present the results for four standard networks for single link failure. However, the presented results are typical of all our simulations. The results are presented for ARPANET (Fig. 3), NJ LATA (Fig. 4), National (Fig. 5) and the European Cost239 (Fig. 6) networks. We ran our algorithms on these networks both with uniform link capacities and with randomly chosen non-uniform link capacities. In the non-uniform case the link capacities range from 20 to 40. We use, as a benchmark, the solution to a linear program that models our problem without the constraint on the number of bypass tunnels or their minimum bandwidth requirement. Note that since the linear program models a problem with fewer constraints, its optimal solution is a lower bound on the optimal solution to our problem. Our main observations are summarized in Table I. As an example, we describe these results for the NJ LATA network. This network has 11 nodes (column II) and 23 links (column III). For the uniform link capacity case, we normalize the results so that each link's capacity is exactly one unit. In this case, the LP (a lower bound on the optimal solution) reserves 6 (column VII) out of the 23 units of total link capacity for protection. The algorithm based on the 2-edge connected subgraph, when optimized with the DFS tree approach, finds an optimal solution of total protection capacity 6 (column VI) units, which is 26% of the total link capacity. Recall that any ring-based approach (e.g., SONET) would reserve at least 50% of the capacity for protection. The other non-optimized algorithm based on the 2-edge connected subgraph also obtains an optimal solution (column V). The tree algorithm reserves 10 (column IV) units of protection capacity (43.5% of the total link capacity and at most 1.67 times the amount reserved by the optimal solution). For non-uniform capacities, the optimal solution reserves at least 188 (column XI) out of a total 610 (column VIII) units of capacity for protection. The algorithm based on the 2-edge connected graph finds a solution which reserves 236 (column X) units of protection capacity (which is at most 1.26 times the optimal solution). For the tree-based algorithm, this number is 306 (column IX), which is at most 1.63 times the optimal solution. In summary, for the uniform capacity case, the optimized 2-edge algorithm has a solution comparable to the optimal algorithm. The 2-edge algorithm, in general, finds a solution that is at most 1.5 times the optimal solution. The tree-based algorithm finds a solution that ranges approximately between 1.6 to 1.9 times the optimal solution.

Effect of hop length: In the algorithms and simulation results presented in this paper, we do not put any restriction on the hop length of the backup tunnels. In the simulation, some of the paths were found to have hop length of 6 or more. The savings were less when we restricted the path length. For example, in NJ LATA network, LP gave a protection capacity of 6 for the uniform case, but gave a protection capacity of 7.3, 6.4, and 6.3

TABLE I
RESULTS FROM REAL-LIFE NETWORKS

			Uniform				Non Uniform			
Network	Node count	Link count	Tree	2-Edge	2-Edge-DFS	LP	Total	Tree	2-Edge	LP
ARPANET	20	32	19	14.5	12	10	914	616	489	343
NJ LATA	11	23	10	6	6	6	610	306	236	188
National	24	44	23	18.5	13	12	1324	770	562	415
Cost 239	11	26	10	9.5	5.5	5.5	710	324	254	186

for hop limit restriction of 4, 5, and 6, respectively. For studying the effect of hop length on the protection capacity, algorithms that restrict the hop length need to be developed. This is left for future study.

Comparison With Existing Work: We now compare the performance of our algorithms with those of other heuristics available in the literature for link-restoration-based protection. We selected existing works that benchmark the performance of their heuristic on the same topologies as used in this paper. For reference purposes we use the "Normalized Spare Capacity Cost (NC)" measure from [21]. This is defined as the ratio of the spare (protection) capacity to the working capacity expressed in percentage. For example, for the NJ LATA network, the NC values for the uniform case for the tree and 2-edge algorithms and the LP are 76.9%, 35.3%, and 35.3%, respectively. For the non-uniform case, the corresponding values are 99.34%, 63.1%, and 44.5%, respectively.

In the work of Herzberg *et al.* [10], heuristics are developed for minimizing the spare capacity needed for guaranteeing link restoration when hop bounds are imposed on the restoration routes. Their work differs from ours in the following respect. First, they do not impose any bounds on the number of restoration paths on which the traffic can be rerouted upon a link failure. Second, in their model the working capacity is pre-determined and links have no capacity bound. Rather, they assume a cost per unit of spare capacity per link. Finally, they impose bounds on the number of hops in the restoration paths. Herzberg *et al.* [10] used the NJ LATA network for evaluating the performance of their heuristic. For this network, for 4 hop bounds or more, their heuristic requires solving an LP with at least 500 constraints and 500 variables. For hop bound of 4 or more, their heuristic settles for 625 units of spare protection capacity for 1252 units of working capacity, giving an NC value of 49.9%.

Murakami and Kim [21] consider the problem of optimizing capacity (working and restoration) and flow assignment for a given traffic demand based on link and path restoration. They formulate the problem as a large-scale linear program and develop special mechanisms to deal with its computational intractability. As in [10], no bounds are imposed on the number of restoration paths on which the traffic can be rerouted upon a link failure. Murakami and Kim [21] also use the NJ LATA network for evaluating the performance of their heuristics. They vary the traffic demand from uniform to non-uniform to random and report the NC values for their different heuristics. The NC values are observed in the range 47%–60%.

It follows that our heuristics have comparable performance to the heuristics in [10] and [21]. These heuristics are based on solving LPs and quickly become computationally intractable [21]. The heuristics given in this paper are combinatorial in nature and have low computational complexity. In addition, the heuristics given in this paper can be used to control the number of restoration paths used by rerouted traffic.

IX. EXTENSIONS AND FUTURE WORK

Note that our scheme is mainly designed for dealing with link failures. However, node and SRLG (shared risk link group) failures are also a common occurrence, and fast reroute-based schemes to protect against these failures are also very appealing. We would like to extend our algorithms to node and SRLG failures. Quality of service is becoming important to support real-time services. For these delay-sensitive services, the network latency must be kept small by limiting the number of hops for both the working traffic and the rerouted traffic. This requires that some bounds be imposed on the number of hops for the bypass tunnels. Extending our scheme to be able to limit the number of hops for the bypass tunnels is therefore an important future direction.

X. CONCLUSION

Pre-provisioning of protection capacities and bypass tunnels results in an efficient implementation of guaranteed fast (local) reroute-based shared restoration in mesh networks. We presented efficient approximation algorithms for minimizing the amount of pre-provisioned protection capacities, while supporting at most two bypass tunnels per link for guaranteed fast (local) reroute. With simulations on standard networks, we showed that our algorithms work well in practice. Finally, we also showed how topology updates can be handled in our framework.

REFERENCES

[1] L. Calvignac *et al.*, "A method for an optimized online placement of MPLS bypass tunnels," Internet Draft, draft-leroux-mpls-bypass-placement-00.txt, 2002.

[2] C. Chekuri, A. Gupta, A. Kumar, J. Naor, and D. Raz, "Building edge-failure resilient networks," in *Proc. Ninth Conf. Integer Programming and Combinatorial Optimization (IPCO) 2002*, pp. 439–456.

[3] H. Choi, S. Subramaniam, and H. Choi, "On double-link failure recovery in WDM optical networks," in *Proc. IEEE INFOCOM 2002*, pp. 808–816.

[4] M. Clouqueur and W. D. Grover, "Availability analysis of span-restorable mesh networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 810–821, May 2002.

[5] B. S. Davie and Y. Rekhter, *MPLS: Technology and Applications*. San Mateo, CA: Morgan Kaufmann, 2000.

[6] G. Ellinas and T. E. Stern, "Automatic protection switching for link failures in optical networks with bi-directional links," in *Proc. IEEE Globecom'96*, pp. 152–156.

[7] W. D. Grover, "Case studies of survivable ring, mesh and mesh-arc hybrid networks," in *Proc. IEEE Globecom'92*, pp. 633–638.

[8] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network reconfiguration," in *Proc. ICC*, 1998, pp. 537–543.

[9] D. Haskin and R. Krishnan, "A method for setting an alternative label switched path to handle fast reroute," Internet Draft, draft-haskin-mpls-fast-reroute-05.txt, 2000.

[10] M. Herzberg, S. J. Bye, and A. Utano, "The hop-limit approach for spare-capacity assignment in survivable networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 6, pp. 775–784, Dec. 1995.

[11] R. R. Iraschko, M. H. MacGregor, and W. D. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks," *IEEE/ACM Trans. Netw.*, vol. 6, no. 3, pp. 325–336, Jun. 1998.

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

[13] S. Kini *et al.*, "Shared backup label switched path restoration," Internet Draft, draft-kini-restoration-shared-backup-01.txt, 2001.

[14] M. S. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proc. IEEE INFOCOM 2000*, pp. 902–911.

[15] M. S. Kodialam and T. V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *Proc. IEEE INFOCOM 2001*, pp. 376–385.

[16] M. Kodialam, T. V. Lakshman, and S. Sengupta, "A simple traffic independent scheme for enabling restoration oblivious routing of resilient connections," in *Proc. INFOCOM 2004*, pp. 2329–2340.

[17] S. S. Lumetta and M. Medard, "Towards a deeper understanding of link restoration algorithms for mesh networks," in *Proc. IEEE INFOCOM 2001*, pp. 367–375.

[18] S. S. Lumetta, M. Medard, and T. Yung-Ching, "Capacity versus robustness: A tradeoff for link restoration in mesh networks," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 1765–1775, Dec. 2000.

[19] M. Medard, S. G. Finn, and R. A. Barry, "WDM loop-back recovery in mesh networks," in *Proc. IEEE INFOCOM 1999*, pp. 752–759.

[20] M. Medard, R. A. Barry, S. G. Finn, W. He, and S. Lumetta, "Generalized loop-back recovery in optical mesh networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 1, pp. 153–164, Feb. 2002.

[21] K. Murakami and H. S. Kim, "Optimal capacity and flow assignment for self-healing ATM network based on line and end-to-end restoration," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 207–221, Apr. 1998.

[22] P. Pan *et al.*, "Fast reroute techniques in RSVP-TE," Internet Draft, draft-ietf-mpls-rsvp-lsp-fastreroute-02.txt, 2003.

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

[24] D. A. Schupke, "The tradeoff between the number of deployed p-cycles and the survivability to dual fiber duct failures," in *Proc. ICC*, 2003, pp. 1428–1432.

[25] D. A. Schupke, A. Autenrieth, and T. Fischer, "Survivability of multiple fiber duct failures," presented at the 3rd Int. Workshop on the Design of Reliable Communication Networks (DRCN), Budapest, Hungary, Oct. 2001.

[26] P. D. Seymour, "Sums of circuits," in *Graph Theory and Related Topics*, J. A. Bondy and U. R. S. Murty, Eds. San Diego, CA: Academic Press, 1979, pp. 341–355.

[27] J. Shi and J. P. Fonseka, "Hierarchical self-healing rings," *IEEE/ACM Trans. Netw.*, vol. 3, no. 12, pp. 690–697, Dec. 1995.

[28] C. Thomassen, "On the complexity of finding a minimum cycle cover of a graph," *SIAM J. Comput.*, vol. 26, no. 3, pp. 675–677, Jun. 1997.

[29] J. P. Vasseur *et al.*, "Traffic engineering fast reroute: Bypass tunnel path computation for bandwidth protection," Internet Draft, draft-vasseur-mpls-backup-computation-02.txt, 2003.

[30] J. Zhang, K. Zhu, and B. Mukherjee, "A comprehensive study on backup reprovisioning to remedy the effect of multiple-link failures in WDM mesh networks," in *Proc. IEEE ICC*, 2004, pp. 1765–1775.

Mansoor Alicherry received the B.Tech. degree from Regional Engineering College, Calicut, India, in 1997, and the M.E. degree from the Indian Institute of Science, Bangalore, India, in 2000, both in computer science.

He is currently a Member of Technical Staff at the Network Software Research Department, Bell Laboratories, Lucent Technologies, Murray Hill, NJ. His interests are primarily in network security and network design algorithms.

Randeep Bhatia recieved the B.Tech. degree in computer science and engineering from Indian Institute of Technology, Delhi, India, the M.S. degree in mathematics and computer science from the University of Illinois at Chicago, and the Ph.D. degree in computer science from the University of Maryland, College Park.

He is with Bell Laboratories, Lucent Technologies, Murray Hill, NJ, working on network design, traffic engineering, and scheduling algorithms. His current research interests are in the area of QoS for emerging multimedia services in next-generation data networks.