Restoration Algorithms for Virtual Private Networks in the Hose Model

Giuseppe F. Italiano, Rajeev Rastogi , Bülent Yener

Abstract— **A *Virtual Private Network* (VPN) aims to emulate the services provided by a private network over the shared Internet. The endpoints of a VPN are connected using abstractions such as Virtual Channels (VCs) of ATM or Label Switching Paths (LSPs) of MPLS technologies. Reliability of an end-to-end VPN connection depends on the reliability of the links and nodes in the fixed path that it traverses in the network. In order to ensure service quality and availability in a VPN, seamless recovery from failures is essential. This work considers the problem of fast recovery in the recently proposed VPN *hose* model. In the hose model bandwidth is reserved for traffic aggregates instead of pairwise specifications to allow any traffic pattern among the VPN endpoints. This work assumes that the VPN endpoints are connected using a *tree* structure and at any time, at most one tree link can fail (i.e., single link failure model). A restoration algorithm must select a set of *backup* edges and allocate necessary bandwidth on them in advance, so that the traffic disrupted by failure of a *primary* edge can be re-routed via backup paths. We aim at designing an optimal restoration algorithm to minimize the total bandwidth reserved on the backup edges. This problem is a variant of optimal graph augmentation problem which is NP-Complete. Thus, we present a polynomial-time approximation algorithm that guarantees a solution which is at most 16 times of the optimum. The algorithm is based on designing two reductions to convert the original problem to one of adding minimum cost edges to the VPN tree so that the resulting graph is 2-connected, which can be solved in polynomial time using known algorithms. The two reductions introduce approximation factors of 8 and 2, respectively, thus resulting in a 16-approximation algorithm with polynomial time complexity.**

I. Introduction

Traditionally, a private network (PN) is established by leased-lines connecting the PN sites (e.g., campuses or branch offices of an enterprise) over a WAN. Since the lines are dedicated, security and bandwidth guarantees are ensured. As the Internet becomes a commercial infrastructure, not only the number of endpoints of a PN gets increased but also the endpoints get geographically dispersed. Thus, connecting a large number of dispersed PN sites with dedicated lines becomes expensive. As a result, in recent years there has been much interest in offering Virtual Private Network (VPN) services over the public Internet. The main challenge has been to provide performance guarantees comparable to private WANs without dedicated leased-lines. The first generation IP-based VPNs technology has mainly focused on security (e.g., IPSEC [1], and tunneling based routing (e.g., L2TP [2]) and fell short of providing any Quality of Service (QoS) guarantees. However, the recent emergence of IP technologies like MPLS [3], Diffserv and RSVP enhances the Internet infrastructure to provide services beyond the traditional best effort. Thus the problem of provisioning VPN services with QoS guarantees has become an active area of research.

There are two popular models for providing QoS in the context of VPNs - the "pipe" model and the "hose" model [3], [4]. In the pipe model, the VPN customer specifies QoS requirements between every pair of VPN endpoints. Thus, the pipe model requires the customer to know the complete traffic matrix, that is, the load between every pair of endpoints.

However, as the number of endpoints grow and as the connectivity dynamics increase, it may be difficult to obtain pairwise statistical bandwidth requirements between the endpoints. Thus, algorithms for establishing VPNs must resort to models with *aggregate* bandwidth demands such as the "hose" model proposed in [4]. In the hose model each VPN site specifies its aggregate ingress and egress bandwidth requests. The ingress bandwidth for an endpoint specifies the incoming traffic from all the other VPN endpoints into the endpoint, while the egress bandwidth is the amount of traffic the endpoint can send to the other VPN endpoints. The hose model is scalable since the customer manages the allocated bandwidth at per flow basis at the network edge while the VPN provider is concerned with only the flow aggregates inside the network. Several provisioning algorithms for VPNs in the hose model have been proposed [3], [4], [5], [6]. Provisioning a VPN requires identifying a subgraph to connect the VPN endpoints, and reserving the necessary bandwidth on the physical links that are used by this subgraph. It is shown that a tree is the optimum topology when the ingress and egress bandwidth requests are symmetrical [5].

Failure of any edge in a VPN tree would disrupt the service unless a backup path was established to reconnect tree. A *restoration algorithm* selects a set of *backup* paths and allocates necessary bandwidth on them in advance, so that the traffic disrupted by failure of a *primary* edge can be re-routed via backup paths.

This work presents restoration algorithms for maintaining a VPN tree in the hose model with symmetric bandwidth requests under transient link failures. Namely, we assume that a link failure in the network is repaired before the next one is presented, which seems realistic in many situations. A naive approach for restoration would be to build a pair of edge-disjoint VPN trees so that if the primary tree gets disconnected then the backup tree would be used. However, this approach would be wasteful under a single link failure model since a backup path can be used to recover from the failure of multiple primary links. Thus a restoration algorithm must consider *sharing* of the backup paths. However, as we show later bandwidth reservation on the backup paths complicates the problem further than simply minimizing

Dipartimento di Informatica, Sistemi e Produzione, Università di Roma "Tor Vergata", Rome, Italy. Email: italiano@disp.uniroma2.it. URL: http://www.info.uniroma2.it/~italiano. Part of this work was done while visiting Columbia University and Bell Laboratories, Lucent Technologies. Partially supported by the IST Programme of the EU under contract n. IST-1999-14.186 (ALCOM-FT), by the Italian Ministry of University and Scientific Research (Project "Algorithms for Large Data Sets: Science and Engineering") and by CNR, the Italian National Research Council under contract n. 00.00346.CT26.

Bell Laboratories, Lucent Technologies 600 Mountain Avenue, Room Murray Hill, NJ 07974, email: rajeev@research.bell-labs.com

Information Sciences Research Center Bell Laboratories, Lucent Technologies 600 Mountain Avenue, Room 2T-314 Murray Hill, NJ 07974, email: yener@research.bell-labs.com

the number of links used in the backup paths.

There are several relevant results on the restoration problem in MPLS capable networks. In [7] restoration algorithms for k link failures, based on concatenation of $k+1$ shortest paths, are presented. In [8], [9], [10] authors present restoration algorithms for LSPs in MPLS enabled networks. Motivated by the single link failure model their restoration objective is based on backup path sharing.

This work complements and advances these results. First, we introduce several cost functions and show the trade-offs among them. The cost functions include minimizing the total bandwidth reserved on the backup paths, minimizing the disruption in the tree links, minimizing the total additional bandwidth reservation needed in the network. Next, we focus on the objective function to minimize total bandwidth on the backup paths. We call this problem optimal augmentation of a VPN tree and note that it is a variant of the optimal graph augmentation problem which is NP-Complete. Thus we present a polynomial time approximation algorithm which gives solutions that are provably at most 16 times the optimum.

Our approach is based on reducing the optimal augmentation problem to the edge connectivity augmentation problem [11], [12] in two phases. In the first reduction, we start from the original graph G and produce a graph G' which has no complications arising from path sharing. G' is obtained from G by replacing entire backup paths with disjoint backup "edges": this costs us an approximation factor of 8. Finding an optimal augmentation for G' is still difficult. We then reduce G' to another graph $\hat{G}$ containing only certain types of non-tree links, and such that the cost of each backup link can be computed more easily: this costs us another approximation factor of 2.

The remainder of the paper is organized as follows. In Section 2, we define our model and the basic properties of the augmentation problem. In Section 3, we introduce the restoration algorithm and its approximation factor via a sequence of reductions. In Section 4 we conclude by summarizing the results.

II. Model and Definitions

We are given an undirected graph $G = (V, E)$ with a set of terminals $W \subseteq V$ that wish to establish communication among each other. Let n and m denote the number of nodes and links in G. We assume that each terminal $i \in W$ has an upper bound B_i on the amount of traffic that can be either sent (egress bandwidth) or received (ingress bandwidth) by i at any point, i.e., for each terminal the ingress bandwidth equals the egress bandwidth. A valid traffic matrix D on W is an assignment of a demand $d_{i,j}$ to each pair of terminals that respects the upper bounds: i.e., for any i we have that $\sum_j d_{i,j} \leq B_i$ and $\sum_j d_{j,i} \leq B_i$. We are also given a VPN tree $T \subseteq G$ that is able to support any set of traffic demands respecting those upper bounds. Namely, each tree edge $e \in T$ has a bandwidth reservation b_e such that the demands corresponding to any valid traffic matrix D can be routed along T. In other words, let $\pi_{i,j}$ be the tree path between terminals i and j: then

$$\sum_{i,j:e\in\pi_{i,j}} d_{i,j} \leq b_e.$$

Edges in the VPN tree are called *primary edges* and their reserved bandwidth is called *primary bandwidth*.

In this paper, we address the problem of maintaining a VPN tree in the transient link failure model. Namely, we assume that network links can fail, but a link failure is repaired before the next one is presented. Our approach is based on choosing a set of *backup* paths to cope with the failure of *any* primary link. Namely, we wish to select a set of backup paths, and allocate *backup bandwidth* on those paths, so that when a primary link e fails, the traffic demands routed on e can be re-routed on the backup paths.

Before defining our objectives in more detail, we need a few more definitions. For the sake of clarity, Table 1 summarizes the notation used throughout the paper.

Definition 1: Given a tree T, and an edge $e = (u, v) \in T$, let T_u and T_v be the two trees obtained after deleting e, with $u \in T_u$ and $v \in T_v$. Let B_{T_u} and B_{T_v} be the sums of (ingress) bandwidths for the terminals in the two trees, i.e., $B_{T_u} = \sum_{i\in W\cap T_u} B_i$ and $B_{T_v} = \sum_{i\in W\cap T_v} B_i$. Then the *bandwidth requirement for edge* e is $b_e = \min\{B_{T_u}, B_{T_v}\}$.

Consider Figure 2, which depicts a VPN tree having three subtrees (shown as triangles) containing the VPN nodes in W. The subtree rooted at node x has total 10 units [1] of ingress and 10 units of egress aggregate bandwidth requirements. The bandwidth requirement for edge $e = (x, a)$ in this figure is 20 which is the minimum of $\{20, 22\}$. Similarly the bandwidth needed on edge (b, c) is 14 which is the minimum of $\{28, 14\}$.

Let $f = (u, v)$ be an edge in $G - T$ such that $u, v \in T$. Inserting f into T will create a fundamental cycle which will include a set $P(f)$ of primary edges. Note that deleting an edge e in $P(f)$ will still induce a new tree $T' = T - e + f$ connecting nodes in W. Thus, if edge e is deleted from T, then the new tree $T' = T - e + f$ could be used to route the traffic demands, provided that there is enough bandwidth reservation in T'. We call f as a **candidate backup edge** for the links in $P(f)$. Since a primary edge can occur in multiple fundamental cycles, it may have multiple candidate backup edges. Thus, an edge $f \in G-T$ becomes a **backup edge** to cover only a subset $\mathcal{P}(f) \subseteq P(f)$ of the edges. For example edge (c, y), shown with a dashed line, is a backup edge for the tree edges (c, d) and (d, y). In the more general case a **backup path** $\pi_e \in G - T$ can be used to obtain such a cycle and each edge in the backup path will be a candidate backup edge. In this work we consider how to choose minimum cost backup paths $\pi_e \in G - T$ for each $e \in T$, and to reserve *backup bandwidth* such that $T'(\pi_e) = (T \cup \pi_e) - \{e\}$ is able to route the demands corresponding to any valid traffic matrix D. Next we define this requirement precisely:

Definition 2: Let $G = (V, E)$ be a graph and let $T \subset G$ be a tree. An *augmentation* for T in G is a set of edges $A_T \subseteq E(G)$ such that the following is true:

- $T \cup A_T$ is 2-edge-connected.
- if $f \in A_T$ covers $e \in T$ (i.e., $e \in \mathcal{P}(f)$), then edges of $T' = T - e + f$ have enough bandwidth reservation so that the demands corresponding to any valid traffic matrix D can be routed along T'.

[1] Without loss of generality, the unit of bandwidth is considered to be Mbps and omitted henceforth.

Symbol	Description
$G =< V, E >$	Graph with nodes V and *bidirectional* edges E
$V(G), E(G)$	node and edge set of graph G, respectively
$e = (u, v)$	edge e that connects nodes u and v
$W \subseteq V$	set of VPN nodes
$D_{\|W\| \times \|W\|}$	pairwise traffic matrix such that $\forall i \in W \sum_j d_{i,j} \leq B_i$ and $\sum_j d_{j,i} \leq B_i$
$T \subseteq G$	a VPN tree that connects the nodes in set W
T_u, T_v	subtrees obtained from deleting primary edge $e = (u, v) \in T$
B_i	bound on the aggregate bandwidth request of node $i \in W$
B_{T_u}, B_{T_v}	total bandwidth in T_u and T_v: $B_{T_u} = \sum_{i \in W \cap T_u} B_i$ and $B_{T_v} = \sum_{i \in W \cap T_v} B_i$
b_e	amount of bandwidth reserved on $e \in E(T)$: $b_e = \min\{B_{T_u}, B_{T_v}\}$
$\pi_T(i, j)$	the unique path on T between nodes i and j
$\pi(i, j)$	the backup path to connect i and j if any link $e \in \pi_T(i, j)$ fails
$\pi_e \in G - T$	backup path for a primary edge $e \in T$
$B(\pi_e)$	bandwidth reservation on the backup path π_e
$T'(\pi_e)$	$(T \cup \pi_e) - \{e\}$
$P(f)$	set of edges in T for which $f \in G - T$ is a *candidate* backup edge
$\mathcal{P}(f)$	set of edges in T for which $f \in G - T$ is a backup edge (i.e., $f \in \pi_e$ for $e \in T$)
$B(f)$	required bandwidth reservation on f: $B(f) = \max_{e \in \mathcal{P}(f)}\{B(e)\}$
$B(e)$	new bandwidth reservation needed on edge $e \in T \cap T'(\pi_e)$
δb_e	$\|B(e) - b_e\|$ for $e \in T \cap T'(\pi_e)$
A	an augmentation of T in G: $A = \{f \in G - T \| \exists \mathcal{P}(f) \forall e \in T\}$
A^*	optimal augmentation of T
$w(\pi_e)$	cost of backup path π_e
$w(A)$	cost of A (i.e., total bandwidth needed by A) : $w(A) = \sum_{f \in A} B(f)$

Fig. 1. Notation Used in the Paper

For the sake of notational simplicity, we will omit the subscript in A_T whenever there is no danger of ambiguity.

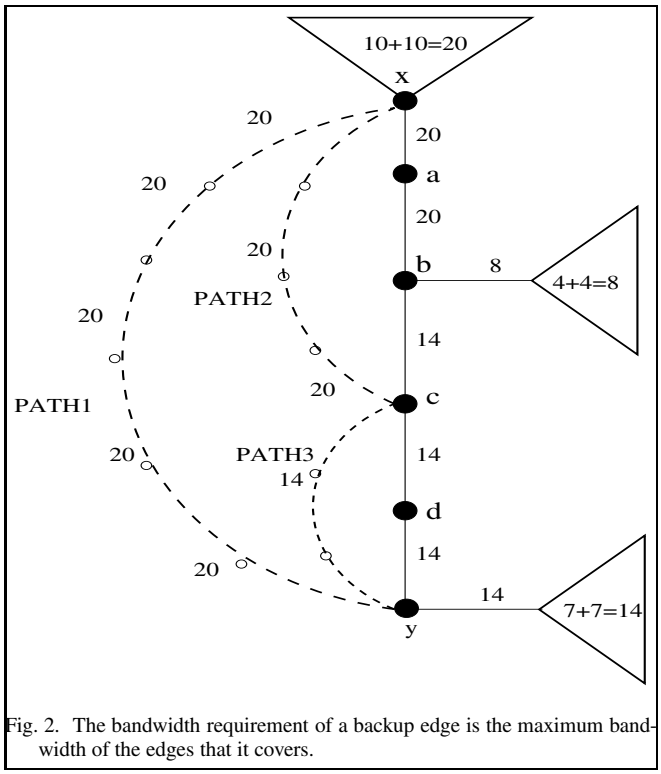

Fig. 2. The bandwidth requirement of a backup edge is the maximum bandwidth of the edges that it covers.

Fact 1: Let T be a tree, and let G be a graph such that $V(T) \subseteq V(G)$ and $E(T) \subseteq E(G)$. Let A be an augmentation for T in G. Let f be an edge in A, let $B(f)$ be the bandwidth requirement of f (i.e., the bandwidth reservation needed for f), and let $\mathcal{P}(f)$ be the set of tree edges for which f is a backup edge. Then

$$B(f) = \max_{e \in \mathcal{P}(f)} \{b_e\}$$

For example consider the backup path PATH2 between nodes x and c in Figure 2 that covers the links $(x, a), (a, b), (b, c)$. To meet the traffic demands, any link f in this path will require bandwidth reservation equal to the maximum of the edges in set $\mathcal{P}(f)$ which is 20.

A. Cost Function for Augmentation and its Variants

We now list several cost measures that can be considered for an optimal augmentation.

A.1 Cost Function CF 1

One simple special case is to consider an augmentation using the minimum number of links. Then, our optimization problem is an instance of the *unweighted 2-edge-connectivity problem* which is known to be NP-Complete. Khuller and Vishkin [13] provide an algorithm with approximation factor of 1.5.

A.2 Cost Function CF 2

In this case, we wish to find an augmentation such that the backup bandwidth reserved on edges in the augmentation is minimum. In other words, we are interested in the optimal augmentation A which minimizes the quantity

$$w(A) = \sum_{f \in A} B(f)$$

Note that this is more difficult than *weighted 2-edge-connectivity* (see Frederickson and Ja'Ja' [14] and Khuller and Thurimella [11]). Indeed, in weighted 2-edge-connectivity, the cost of a non-tree edge f is given, while here it depends on which edges are covered by f, i.e., on $\mathcal{P}(f)$.

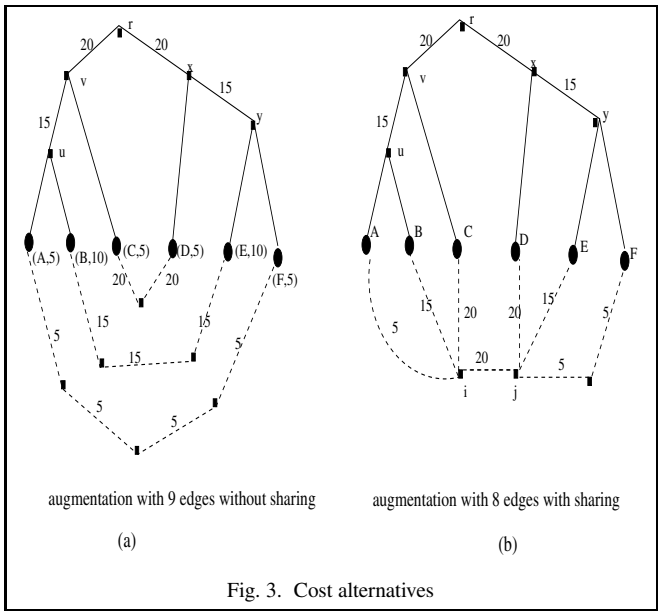

Fig. 3. Cost alternatives

A.3 Cost Function CF 3

We define a more precise cost function for a backup path which has two components:

(1) the total bandwidth required on the links of the backup path (**backup edge costs**);
(2) the total additional bandwidth required on the primary edges (**primary edge costs**).

We denote the new *total bandwidth* reservation needed on a primary edge e by $B(e)$. Let $\delta b_e = |B(e) - b_e|$ be the primary edge cost (i.e., additional bandwidth required) for $e \in T$. Essentially, $B(e)$ is the maximum bandwidth reserved on primary edge e in all the trees $T'(\pi_{e'}) = (T \cup \pi_{e'}) - \{e'\}$, for primary edges e'. (Note that $\pi_{e'}$ denotes the backup path for edge e').

For example, there are two choices in Figure 2 to cover the links on the tree path between x and y. We can choose PATH1, or PATH2 and PATH3 together. If we choose the former, each $f \in$ PATH1 will have $B(f) = 20$ yielding a total cost of $6 \times 20 = 120$ in the backup path. The choice of a backup path may require increasing the bandwidth on the primary edges as well. For example suppose that link $e = (a, b) \in T$ fails. The maximum bandwidth on this link is set to 20 which is the total traffic to and from subtree rooted under x. The new path to x in $T'(\pi_e) = (T \cup \pi_e) - \{e\}$ is via the tree links $(b, c), (c, d,), (d, y)$ each of which needs additional 20-14 =6 units of bandwidth. Thus the total bandwidth is 120+18= 138. If PATH2 and PATH3 are used the total cost would be $20 \times 4 + 14 \times 3 + 6 = 128$. Thus we can define a cost measure for a backup path as a linear combination of the two components explained above.

Definition 3: The cost of an augmentation A taking into account bandwidth reservations on both primary as well as backup edges is given by:

$$w(A) = \sum_{f \in A} B(f) + \sum_{e \in T} \delta b_e$$

We remark that there are several trade-offs between these two components, depending on the problem considered. For instance, consider the example in Figure 3 which shows two augmentation (dashed lines) of the same VPN tree (solid lines) with VPN nodes and their bandwidth requests shown in the parenthesis. Backup path PATH1 in Figure 3.a creates a cycle that includes the links $(A, u), (u, v), (v, r), (r, x), (x, y), (y, F)$. However, the backup links in this path are used to cover the links (A, u) and (y, F) (i.e., $\forall f \in$ PATH1, $\mathcal{P}(f) = \{(A, u), (y, F)\}$). The cost of the first augmentation is $4 \times 5 + 3 \times 15 + 2 \times 20 + (5 + 15 + 15 + 5) = 145$ where the term in parenthesis is the additional bandwidth required for the tree links. The second augmentation shown in Figure 3.b uses one less link by sharing (i, j) among all the augmentation paths and has the same total cost of 145. However suppose that VPN nodes C and D increase their bandwidth request to 10 then the cost of augmentation without sharing a backup edge would be 155 while for the augmentation with sharing it would be 160. Thus, it is not always desirable to *share the backup links* to obtain an optimal augmentation of a VPN tree.

For the sake of succinctness, in the remainder of the paper we will focus primarily on cost function **CF 2**: i.e., an augmentation for which $w(A) = \sum_{f \in A} B(f)$ is minimum over all augmentations A_T of T. Our algorithm for cost function **CF 2** can be extended with more sophisticated techniques to the case of cost function **CF 3**. Details of these extensions will be presented in the full version of the paper.

III. Approximation Algorithms for an Optimal Augmentation

In this section, we present an algorithm to find a 16-approximation to the optimal augmentation problem for cost function **CF 2**. Our algorithm is based on a sequence of reductions. The high-level ideas behind those reductions are as follows. In the first reduction, we start from G and produce a graph G' which has no complications arising from path sharing. G' is obtained from G by replacing entire backup paths with disjoint backup "edges": this will cost us an approximation factor of 8. Finding an optimal augmentation for G' will still be difficult: in particular, as we discussed above, the cost of a backup edge f in G' still depends on the tree edges that are covered by f. We will reduce G' to another graph $\hat{G}$, such that the cost of each backup edge is fixed and can be computed more easily: this will cost us another approximation factor of 2. In order to compute the optimal augmentation for T in $\hat{G}$, we will use the 2-edge-connectivity augmentation algorithm of Khuller and Thurimella [11]. We will now see those reductions more in detail.

A. Constructing graph G'

We will start by showing that an optimal augmentation is given by a forest of trees:

Lemma 1: Let T be a tree, and let G be a graph such that $V(T) \subseteq V(G)$ and $E(T) \subseteq E(G)$. Then an optimal augmentation for T in G is acyclic.

Proof: Let A^* be an optimal augmentation for T in G. Assume by contradiction that there is a cycle λ in A^*. Without loss of generality, λ is a simple cycle. For each edge $f \in A^*$, let $B(f)$ be its bandwidth requirement, and let $\mathcal{P}(f)$ be the set of tree edges for which f is a backup edge. Let $f' \in A^*$, be the edge with minimum bandwidth requirement in λ, i.e., $B(f') =$

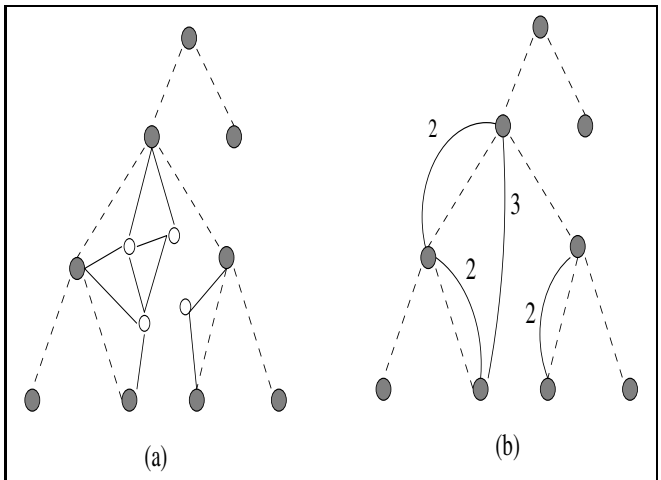

Fig. 4. Constructing G' from G. The graph G is depicted in (a) and the corresponding graph G' is shown in (b).

$\min_{f \in \lambda}\{B(f)\}$. We claim that if f' is deleted from A^*, then the other edges of λ can be used as backup paths for $\mathcal{P}(f')$ without having their bandwidth requirements increased. Indeed, let e be any tree edge in $\mathcal{P}(f')$, and let $\pi(e) \subseteq \lambda$ be the maximal subpath of λ containing f' used by the backup path π_e of e in A^*: note that $\pi(e)$ is always nonempty as $f' \in \pi(e)$. Then, after deleting f', we can replace $\pi(e)$ by $\lambda - \pi(e)$ in the backup path of e. Since $B(f') \leq B(f)$ for each $f \in \lambda$, the bandwidth requirement of edges in $\lambda - \pi(e)$ will not be affected by this change. In summary, $A' = A^*/\{f'\}$ is still an augmentation for T in G, having $w(A') = w(A^*) - B(f') < w(A^*)$. This contradicts the optimality of A^*. ■

Lemma 1 states that the edges in the optimal augmentation induce a forest. Next, we show how to exploit this property for our first reduction.

Definition 4: Let T be a tree, and let G be a graph such that $V(T) \subseteq V(G)$ and $E(T) \subseteq E(G)$. Let X be defined as follows: $V(X) = V(T)$ and there is an edge (u, v) in X if and only if there is a path from u to v in $G - T$ (i.e., a path in G avoiding edges of T). Let $G' = T \cup X$ be such that $V(G') = V(T)$ and $E(G') = E(T) \cup E(X)$. Thus, G' only contains vertices from T. Further, each non-tree edge $f' = (u, v)$ in G' has a weight $w_{f'}$ which is the number of edges in $\pi(u, v)$, the shortest path (i.e., the path with minimum number of edges) between vertices u and v in $G - T$.

Figure 4.b illustrates the graph G' constructed for the graph G depicted in Figure 4.a. In the figure, the vertices of T are shaded and tree edges are drawn using dotted lines. As shown in Figure 4.b, G' only contains vertices from T; the weights of non-tree edges in G' are used to label the edges.

An augmentation A' in G' consists of non-tree edges that cover all the tree edges; each non-tree edge $f' = (u, v)$ in A' can serve as a backup edge for any tree edge e in the unique path between u and v in T. Thus, the cost (for **CF 2**) of an augmentation A' in G' is given as

$$w(A') = \sum_{f' \in A} w_{f'} \cdot B(f')$$

where A' is a set of non-tree edges in G' and $B(f') = \max_{e \in \mathcal{P}(f')}\{b_e\}$. In the remainder of this subsection, we show that there is an augmentation A' for T in G' whose cost is within a factor of 8 of the optimal augmentation for T in G.

Let A^* be an optimal augmentation for T in G. Suppose we assign each edge e in T to bins as follows. If the bandwidth b_e reserved on e satisfies $2^{l-1} < b_e \leq 2^l$, then edge e is assigned to bin l. Let β_l denote the set of tree edges assigned to bin l. Also, let L denote the maximum index for a bin to which a tree edge is assigned; that is, for all tree edges e, $b_e \leq 2^L$. We define A_l^* to be the augmentation consisting of edges in A^* that protect a tree edge in bin l. In other words, if for an edge $f \in A^*$, there exists a tree edge $e \in \mathcal{P}(f) \cap \beta_l$, then $f \in A_l^*$. Furthermore, the bandwidth reserved on each backup edge f in augmentation A_l^* is 2^l. Clearly, for a tree edge $e \in \beta_l$, since A^* contains a backup path for e, augmentation A_l^* must also contain the same backup path for e. Also, since $b_e \leq 2^l$ and the bandwidth reserved on each edge of A_l^* is 2^l, the backup path in A_l^* has sufficient bandwidth to protect e. Thus, each A_l^* covers all the edges in β_l, and the augmentations $A_0^*, \ldots, A_L^*$ protect all the tree edges. Note that the cost of A_l^* is given by $w(A_l^*) = 2^l \cdot |A_l^*|$.

Lemma 2: Let A_l^* be the edges of an optimal augmentation A^* that protect a tree edge in β_l. Then $\sum_l w(A_l^*) \leq 4 \cdot w(A^*)$.

Proof: Consider an edge $f \in A^*$. Let e be the edge in $\mathcal{P}(f)$ with the maximum value of b_e. Thus, $B(f) = b_e$. Suppose that $e \in \beta_i$. Then $2^{i-1} < b_e \leq 2^i$ and f can be present only in augmentations $A_0^*, \ldots, A_i^*$ (since f does not protect edges in $\beta_{i+1}, \ldots, \beta_L$, it does not belong to $A_{i+1}^*, \ldots, A_L^*$). In each augmentation A_l^*, $0 \leq l \leq i$, the bandwidth reserved on edge f is 2^l. Thus, the total bandwidth reserved on f in augmentations $A_0^*, \ldots, A_L^*$ is given by $\sum_{l=0}^{i} 2^l \leq 2^{i+1}$. Since $2^{i-1} < B(f) = b_e \leq 2^i$, it follows that the bandwidth reserved on f in $A_0^*, \ldots, A_L^*$ is $\leq 4 \cdot B(f)$. The lemma follows since the total bandwidth reserved on the edges in augmentations $A_0^*, \ldots, A_L^*$ is $\leq 4 \cdot \sum_{f \in A^*} B(f)$ and $w(A^*) = \sum_{f \in A^*} B(f)$. ■

From the above lemma, it follows that the sum of the costs of augmentations $A_0^*, \ldots, A_L^*$ is at most $4 \cdot w(A^*)$. In the following lemma, we show that for each augmentation A_l^* in G, there exists an augmentation A_l' in G' that protects all the tree edges in β_l and whose cost is within a factor of 2 of $w(A_l^*)$.

Lemma 3: Let A_l^* be the edges of an optimal augmentation A^* in G that protect a tree edge in β_l. Let G' be the graph of Definition 4: then there is an augmentation A_l' that protects edges in β_l in G' such that $w(A_l') \leq 2 \cdot w(A_l^*)$.

Proof: We show how to use the edges of A_l^* to identify a collection of edges in $G' - T$ that form an augmentation A_l' for β_l in G', such that $w(A_l') \leq 2 \cdot w(A_l^*)$.

By Lemma 1, A_l^* is a collection of trees (see e.g., the example in Figure 5.a). Perform a Euler tour on each tree and partition each tour into segments between two nodes in T (see Figure 5.b). Let $\sigma(u, v)$ be a segment in the Euler tour between vertices u and v in T. Note that $\sigma(u, v)$ corresponds to a path in G that does not include any tree edges. Thus, replacing segment $\sigma(u, v)$ with a copy of the shortest path $\pi(u, v)$ between its endpoints does not affect the 2-edge-connectivity between the terminals. So replacing each segment with a copy of the corresponding shortest path in $G' - T$ leaves the terminals still

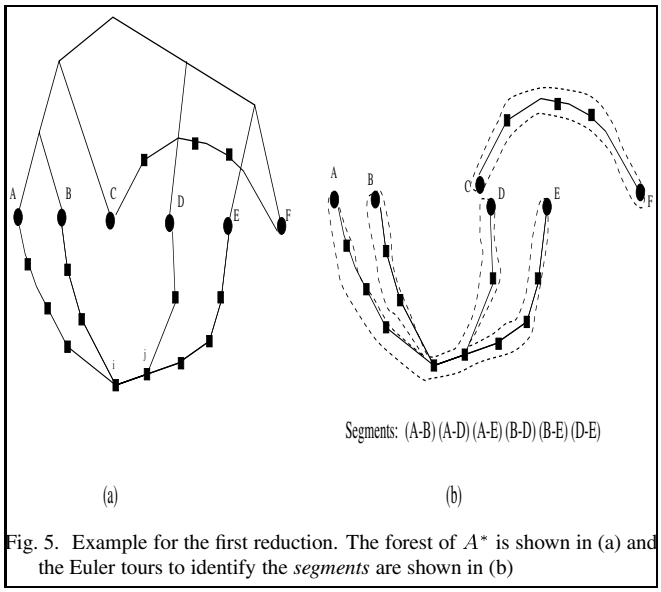

Fig. 5. Example for the first reduction. The forest of A^* is shown in (a) and the Euler tours to identify the *segments* are shown in (b)

2-edge-connected. Furthermore, if we reserve a bandwidth of 2^l on each edge of every shortest path segment, the sum of bandwidth requirements on all those segments is at most $2 \cdot w(A_l^*)$, since each edge of A_l^* appears twice in the Euler tours and the bandwidth reserved on each edge of A_l^* is 2^l. Observe that because augmentation A_l^* covers all the tree edges in β_l, the collection of shortest path segments also protect all the edges in β_l.

Replacing one segment with an edge between its endpoints is equivalent to shrinking a path of degree-two vertices into one edge: once again, this does not affect the 2-edge-connectivity between the terminals. So replacing each segment $\pi(u, v)$ between nodes u, v in T with the corresponding edge $f' = (u, v)$ of $G' - T$ leaves the terminals still 2-edge-connected. Due to Definition 4, we know that $w_{f'}$, the weight of f' in G' is equal to the number of edges in $\pi(u, v)$. Also, $\mathcal{P}(f')$, the set of tree edges for which f' is a backup edge, consists of all the edges in β_l in the unique path between u and v in T. Thus, the bandwidth requirement of edge f' in A'_l, $B(f') = \max_{e \in \mathcal{P}(f')}\{b_e\}$, which can be at most 2^l. As a result, $w_{f'} \cdot B(f')$, the contribution of edge f' to $w(A'_l)$ is at most the bandwidth requirement of segment $\pi(u, v)$, which is equal to 2^l times the number of edges in $\pi(u, v)$. This yields an augmentation A'_l for β_l in G', whose total cost is at most $2 \cdot w(A_l^*)$. ■

From Lemmas 2 and 3 above, it follows that there exist augmentations $A'_0, A'_1, \ldots, A'_L$ in G' that protect all edges of T and such that $\sum_l w(A'_l) \leq 8 \cdot w(A^*)$. Thus, $A' = \cup_l A'_l$ is an augmentation for T in G' whose cost is within a factor of 8 of the optimal augmentation for T in G. Note that, each edge $f' \in A'$ serves as a backup edge for a tree edge e if and only if it serves as a backup edge for e in some A'_l. Thus, the bandwidth reserved on f' in A' is no more than the sum of the bandwidths reserved on it in $A'_0, A'_1, \ldots, A'_L$, and $w(A') \leq \sum_l w(A'_l)$.

We have thus reduced the problem of finding an augmentation for T in G to that of finding one in G', which is the problem we address in the following subsection. Using the algorithm from the next subsection, we will first find an approximate solution for an augmentation A' of T in the graph G' defined as in Definition 4. We expect this augmentation problem to be easier than the original problem, since the edges in G' between vertices of T are disjoint, and thus there are no complications arising from path sharing. Let κ be the approximation factor for this (we will show later that $\kappa = 2$).

Next, we will use this augmentation A' for G' to construct an augmentation of T in G (the original problem). Basically, we will replace each edge $f' = (u, v)$ in A' with the corresponding shortest path $\pi(u, v)$ between nodes u and v in $G - T$. Further, each edge in $\pi(u, v)$ will be a backup edge for all tree edges in $\mathcal{P}(f')$, the set of edges protected by f' in A'. With the help of Lemmas 2 and 3, this will give us an approximation factor of $8 \cdot \kappa = 16$.

B. Finding an augmentation for G'

In this section we will show how to obtain a near-optimal solution to the augmentation problem on G'. For the sake of brevity, we will only sketch here the main ideas underlying the algorithm. Recall that, according to Definition 4, G' consists of tree T plus a set of non-tree edges between pairs of vertices in T. Further, each non-tree edge $f' = (u, v)$ has an associated weight $w_{f'}$, which is essentially the number of edges in $\pi(u, v)$. A non-tree edge $f' = (u, v)$ can serve as a backup edge for any tree edge e in the unique path between u and v in T. Thus, the cost (for **CF 2**) of an augmentation A' in G' is given as

$$w(A') = \sum_{f' \in A} w_{f'} \cdot B(f')$$

where A' is a set of non-tree edges in G' and $B(f') = \max_{e \in \mathcal{P}(f')}\{b_e\}$. Our goal is to compute the augmentation with the minimum cost.

B.1 Choosing root for tree T

Before presenting our algorithm for computing a near-optimal augmentation for G', we show that T contains a vertex $r(T)$ that satisfies the following property: let $e_1, \ldots, e_k$ be the sequence of edges in T from $r(T)$ to any vertex v in T. Then, $b_{e_1} \geq b_{e_2} \cdots \geq b_{e_k}$. We choose $r(T)$ as the root for T.

Recall from Definition 1 that the bandwidth requirement for an edge $e = (u, v) \in T$ is given by $b_e = \min\{B_{T_u}, B_{T_v}\}$. In order to show the above property for node $r(T)$, we construct a directed tree T_{dir} from T by giving a direction to each edge $e = (u, v)$ of T as follows:

- If $B_{T_u} < B_{T_v}$, then direct the edge towards u.
- If $B_{T_v} < B_{T_u}$, then direct the edge towards v.
- If $B_{T_v} = B_{T_u}$, then direct the edge towards the component which contains a particular leaf, say, $\hat{x}$.

Clearly, T_{dir} must contain a node whose indegree is 0 (otherwise, T would contain a cycle) — we choose this node in T_{dir} with no incoming edges as $r(T)$. We show that $r(T)$ is indeed unique and satisfies the above-mentioned property using the following property of T_{dir} in the lemma below [5].

Lemma 4: Every edge e in T_{dir} is directed away from $r(T)$.

Proof: Let $e = (x, y)$ be an edge in tree T such that x is closer to $r(T)$ than y in T. We show that e is directed from x to y in T_{dir}. Consider the path in T from $r(T)$ to x. We know that the first edge (u, v) along the path is directed away from

$u = r(T)$. So, $B_{T_v} \leq B_{T_u}$. Since (u, v) is the first edge of the path from $r(T)$ to x, $T_u \subseteq T_x$ and also, $T_y \subseteq T_v$. Thus, we get, $B_{T_y} \leq B_{T_v} \leq B_{T_u} \leq B_{T_x}$. If $B_{T_y} < B_{T_x}$, then edge e is directed from x to y and we are done. The only other possibility is $B_{T_y} = B_{T_x}$. But then, it must be the case that $T_u = T_x$, $T_v = T_y$ and $B_{T_u} = B_{T_v}$. As a result, it follows that $u = x$ and $v = y$, and since edge (u, v) is directed from u to v, edge (x, y) must also be directed from x to y. ■

Note that from the above lemma, one can easily show that $r(T)$ is unique since every other node in T_{dir} has an edge directed into it (and consequently, an indegree of 1). Further, if $r(T) = x_0, x_1 \ldots, x_k = v$ is a tree path from $r(T)$ to v in T involving edges $e_1 = (x_0, x_1), e_2 = (x_1, x_2), \ldots, e_k = (x_{k-1}, v)$, then, $B_{T_{x_1}} \geq B_{T_{x_2}} \geq \cdots B_{T_{x_k}}$. Since $b_{e_i} = B_{T_{x_i}}$, it follows that $b_{e_1} \geq b_{e_2} \cdots \geq b_{e_k}$. We choose $r(T)$ as the root of T.

B.2 Constructing graph $\hat{G}$

Next, we will form a graph $\hat{G}$ from G' as follows. The rationale for transforming G' to $\hat{G}$ is that in an augmentation A' for T in G', the cost of a backup edge $f' \in A'$ varies depending on the tree edges covered by f'. This makes computing the optimal augmentation in G' difficult. In order to address this problem, each backup edge $\hat{f} = (u, v)$ in the new graph $\hat{G}$ has a fixed cost $c_{\hat{f}}$, and protects all the tree edges along the unique path between u and v. This makes it possible to devise efficient algorithms for computing the optimal augmentation in $\hat{G}$.

In $\hat{G}$, the tree edges in G' are retained without any modifications. However, each non-tree edge in $G' - T$, is replaced by a different set of edges in $\hat{G}$. Consider any edge $f' = (u, v)$ in $G' - T$ between two vertices u and v in T, and let $\mathsf{lca}(u, v)$ denote the least common ancestor of u and v in T. Also, let $u = u_0, u_1, \ldots, u_p = \mathsf{lca}(u, v)$ be the sequence of nodes in T from u to $\mathsf{lca}(u, v)$, and $v = v_0, v_1, \ldots, v_q = \mathsf{lca}(u, v)$ be the sequence of nodes in T from v to $\mathsf{lca}(u, v)$. Then, we perform the following actions for each edge $f' = (u, v)$ in $G' - T'$ to derive $\hat{G}$ from G'.

- Delete edge f' from G'.
- Add edges $\hat{f}_i = (u, u_i)$ for $i = 1, \ldots, p$. Further, assign each edge $\hat{f}_i$ a cost $c_{\hat{f}_i} = w_{f'} \cdot \max_{1 \leq j \leq i}\{b_{(u_{i-1}, u_i)}\}$.
- Add edges $\hat{g}_i = (v, v_i)$ for $i = 1, \ldots, q$. Further, assign each edge $\hat{g}_i$ a cost $c_{\hat{g}_i} = w_{f'} \cdot \max_{1 \leq j \leq i}\{b_{(v_{i-1}, v_i)}\}$.

In the above set of actions, $w_{f'}$ is the weight of edge f' in G' and $b_{(u_{i-1}, u_i)}$ is the bandwidth reserved on tree edge (u_{i-1}, u_i). Note that it is possible that multiple edges may be added between a pair of vertices u and v in $\hat{G}$ – in this case, we only retain the edge (u, v) with the minimum cost, and delete the remaining edges between the vertices.

Figure 6 illustrates the actions for non-tree edge $f' = (u, v)$ shown in bold and having weight $w_{f'} = 2$. In the figure, tree edges are drawn using dotted lines and the bandwidth reservation for each tree edge is placed next to the edge. Thus, $b_{(u,u_1)} = 1$ and $b_{(v,v_1)} = 2$. Constructing $\hat{G}$ from G' involves replacing edge $f' = (u, v)$ with four edges (shown using solid lines), two from u to u_1 and $\mathsf{lca}(u, v)$, and another two from v to v_1 and $\mathsf{lca}(u, v)$. Adjacent to the four new edges in the figure, are depicted their respective costs.

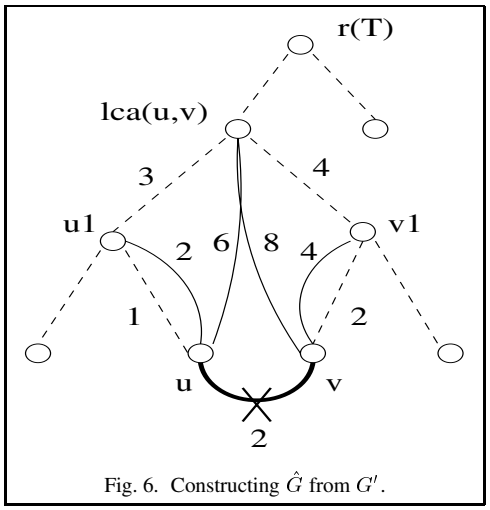

Fig. 6. Constructing $\hat{G}$ from G'.

The above actions produce a graph $\hat{G}$ for which T is a spanning tree, and such that non-tree edges can only be back edges: i.e., if (u, v) is a non-tree edge then either u is an ancestor of v or v is an ancestor of u in T. Further, the cost $c_{\hat{f}}$ of a non-tree edge $\hat{f} = (u, x)$ in $\hat{G}$ (generated due to edge $f' = (u, v)$ in G') is essentially the product of $w_{f'}$ and the maximum bandwidth of tree edges between u and x. Thus selecting edge $\hat{f}$ in $\hat{G}$ is basically equivalent to selecting edge f' in G' as the backup edge for all the tree edges between u and x. Furthermore, since the bandwidth reserved on tree edges is higher for edges closer to the root, the effect of picking any edge f' in G' as a backup edge can be achieved by selecting at most two edges in $\hat{G}$.

An augmentation $\hat{A}$ for T in $\hat{G}$ is a subset of $\hat{G} - T$ and has the property that $T \cup \hat{A}$ is 2-edge-connected; thus, for every tree edge, $\hat{A}$ contains a backup edge. Further, we define the cost of $\hat{A}$ in $\hat{G}$ to be $w(\hat{A}) = \sum_{\hat{f} \in \hat{A}} c_{\hat{f}}$. We can show that the minimum cost augmentation for T in the graph $\hat{G}$ will yield a solution of cost at most twice the optimal in G'.

Lemma 5: Let A' be an augmentation for T in G' with cost $w(A')$. Then, there is an augmentation $\hat{A}$ for T in $\hat{G}$ such that $w(\hat{A}) \leq 2 \cdot w(A')$.

Proof: We will construct from A' an augmentation $\hat{A}$ for T in $\hat{G}$ whose cost is less than or equal to $2 \cdot w(A')$. Suppose that $f' = (u, v) \in A'$ serves as the backup edge for tree edges in $\mathcal{P}(f')$. Note that $\mathcal{P}(f')$ can only contain tree edges along the path between u and $\mathsf{lca}(u, v)$, and v and $\mathsf{lca}(u, v)$. Let $u = u_0, u_1, \ldots, u_p = \mathsf{lca}(u, v)$ be the sequence of nodes in T from u to $\mathsf{lca}(u, v)$, and $v = v_0, v_1, \ldots, v_q = \mathsf{lca}(u, v)$ be the sequence of nodes in T from v to $\mathsf{lca}(u, v)$. Further, let (u_{i-1}, u_i) be the edge in $\mathcal{P}(f')$ with the highest index i along the path from u to $\mathsf{lca}(u, v)$, and (v_{j-1}, v_j) be the edge in $\mathcal{P}(f')$ with the highest index j along the path from v to $\mathsf{lca}(u, v)$. Then, we add to $\hat{A}$ the pair of edges $\hat{f} = (u, u_i)$ and $\hat{g} = (v, v_j)$, both of which are contained in $\hat{G}$. Clearly, $\hat{f}$ is the backup edge for edges in $\mathcal{P}(f')$ contained in the path from u to $\mathsf{lca}(u, v)$, while $\hat{g}$ is the backup edge for edges in $\mathcal{P}(f')$ contained in the path from u to $\mathsf{lca}(u, v)$. Also, $c_{\hat{f}} \leq w_{f'} \cdot b_{(u_{i-1}, u_i)}$ since as we showed earlier, with $r(T)$ as the root of T, it is the case that $b_{(u_{i-1}, u_i)} \geq b_{(u_{i-2}, u_{i-1})} \geq \cdots \geq b_{(u_0, u_1)}$. (Note

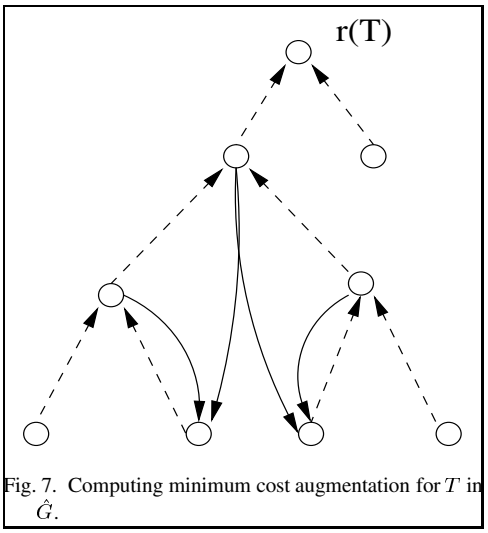

Fig. 7. Computing minimum cost augmentation for T in $\hat{G}$.

that if $\hat{f}$ is generated due to an edge different from f', then $c_{\hat{f}} \leq w_{f'} \cdot b_{(u_{i-1},u_i)}$ since the edge added to $\hat{G}$ due to f' must have been deleted due to its higher cost). Similarly, it can be shown that $c_{\hat{g}} \leq w_{f'} \cdot b_{(v_{j-1},v_j)}$. Furthermore, since $\mathcal{P}(f')$ contains both (u_{i-1},u_i) and (v_{j-1},v_j), $B(f') \geq b_{(u_{i-1},u_i)}$ and $B(f') \geq b_{(v_{j-1},v_j)}$. Thus, it follows that $c_{\hat{f}} + c_{\hat{g}} \leq w_{f'} \cdot (b_{(u_{i-1},u_i)} + b_{(v_{j-1},v_j)}) \leq 2 \cdot w_{f'} \cdot B(f')$, and as a result, $w(\hat{A}) \leq 2 \cdot w(A')$. ■

We can use the above lemma to compute an augmentation for T in G' whose cost is at most 2 times the cost of the optimal augmentation for G'. We achieve this by first computing a minimum cost augmentation for T in $\hat{G}$ using the 2-edge-connectivity augmentation algorithm of Khuller and Thurimella [11] (described in the following subsection). Let $\hat{A}$ denote this optimal augmentation. Clearly, due to Lemma 5, $w(\hat{A})$ is within a factor of 2 of the cost of the optimal augmentation for G'. We now show how we can construct an augmentation A' for G' such that $w(A') \leq w(\hat{A})$. For each edge $\hat{f} = (u,v)$ in $\hat{A}$ that was added to $\hat{G}$ because of edge f' in G', we simply add f' to A'. Edge f' serves as the backup edge in A' for all tree edges between vertices u and v. Thus, f''s contribution to $w(A')$ is the product of $w_{f'}$ and the maximum bandwidth of tree edges between u and v, which is essentially $c_{\hat{f}}$. Thus, $w(A') \leq w(\hat{A})$, and A' has a cost that is at most 2 times the cost of the optimal augmentation for G'.

B.3 Finding Optimal augmentation for $\hat{G}$

We can compute the minimum cost augmentation for T in $\hat{G}$ using the algorithm of Khuller and Thurimella [11], which is as follows.

1. Direct all edges of T in $\hat{G}$ towards $r(T)$, the root of T. Set their cost to zero.
2. For every other edge $\hat{f} = (u,v)$ in $\hat{G} - T$ such that u is an ancestor of v, direct the edge from u to v, and set its cost to $c_{\hat{f}}$.
3. Find a minimum weight branching in the directed graph rooted at $r(T)$. For each directed edge $\hat{f} \in \hat{G} - T$ that is picked as part of the branching, add the (corresponding undirected) edge in $\hat{G}$ to $\hat{A}$.

Figure 7 illustrates Steps 1 and 2 of the algorithm with tree edges depicted using dotted lines and non-tree edges drawn using solid lines. In [11], [12], it is shown that the minimum weight branching from $r(T)$ in the directed graph yields the minimum cost augmentation for T in $\hat{G}$. Thus, putting all the pieces together (Lemmas 2, 3 and 5), yields the following theorem.

Theorem 1: A near-optimal augmentation for VPN tree T in G can be computed with a performance guarantee of 16.

C. Time Complexity

Our overall algorithm for computing a 16-approximation for the optimal augmentation consists of the following three steps:
1. Construct graph G' from G by introducing a single edge between vertices $u, v \in T$ if they are connected in $G - T$.
2. Construct graph $\hat{G}$ from G' by introducing for every edge $(u,v) \in G'$, edges between u/v and nodes on the path from u/v to $\mathsf{lca}(u,v)$.
3. Find the optimal augmentation for T in $\hat{G}$.

Suppose that G has n vertices and m edges. The worst-case time complexity of our algorithm is dominated by the first step which requires shortest paths to be computed in $G - T$ between every pair of tree nodes – this takes $O(n \cdot (m + n \cdot \log n))$ time. The second step can be achieved in time $O(n^2)$ time since $\mathsf{lca}(u,v)$ for n^2 pair of vertices can be found in $O(n^2)$ time using the algorithm of Harel and Tarjan [15]. Finally, the optimal augmentation for T in $\hat{G}$ can be found in $O(n^2)$ time using the minimum weight branching algorithm from [16].

Theorem 2: The worst-case time complexity of our algorithm for computing a 16-approximation for the optimal augmentation for VPN tree T in G is $O(m \cdot n + n^2 \cdot \log n)$.

IV. Conclusions

Given a VPN tree with a bandwidth reservation based on the hose model, we developed novel restoration algorithms for coping with single link failure in the tree. Our approach is based on identifying a set of *backup* edges and reserving bandwidth on them such that service quality is ensured in case of any *primary* edge failure in the tree. We introduced several cost functions in the selection of backup edges and chose the one that aims to minimize the total bandwidth reserved in the backup edges. This problem is a variant of optimal graph augmentation which is known to be NP-Complete. Thus, our main result is an approximation algorithm for minimum cost restoration with a performance guarantee of 16. Since the bound on the approximation ratio is provable, we omit the experimental performance study in this version.

Acknowledgments: We would like to thank Amit Kumar for his comments on earlier drafts of this paper, and for pointing out to us the scaling and rounding technique of Lemma 2.

References

[1] S. Kent and R. Atkinson, "Security architecture for the internet protocol," *RFC 2401*, nov 1998.
[2] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, "Layer two tunneling protocol," *RFC 2661*, August 1999.
[3] B. Davie and Y. Rekhter, *"MPLS Technology and Applications"*, Morgan Kaufmann Publishers, 2000.

[4] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, "A flexible model for resource management in virtual private networks," *Proceedings ACM SIGCOMM*, 1998.

[5] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," in *Proceedings ACM SIGCOMM*, 2001.

[6] A. Gupta, A. Kumar, J. Kleinberg R. Rastogi, and B. Yener, "Provisioning a virtual private network: A network design problem for multicommodity flow," in *Proceedings ACM STOC*, 2001.

[7] A. Bremler-Barr, Y. Afek, H. Kaplan, E. Cohen, and M. Merritt, "Restoration by path concatenation: Fast recovery of mpls paths," *Proceedings of ACM SIGMETRICS*, 2001.

[8] M. Kodialam and T. V. Lakshman, "Dynamic routing bandwidth guaranteed tunnels with restoration," *Proceedings IEEE INFOCOM*, 2000.

[9] M. Kodialam and T. V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," *Proceedings IEEE INFOCOM*, 2001.

[10] S. Kini, M. Kodialam, T. Lakshman, and C. Villamizar, "Shared backup label switched path restoration," *IETF Internet draft, draft-kinirestoration -shared-backup-00.txt*, November 2000.

[11] S. Khuller and R. Thurimella, "Approximation algorithms for graph augmentation," *Journal of Algorithms*, vol. 14-2, pp. 214–225, 1993.

[12] D. S. Hochbaum, *"Approximation Algorithms for NP-Hard Problems"*, PWS Publishing Company, 1997.

[13] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings," *Journal of the ACM*, vol. 41(2), pp. 214–235, 1994.

[14] G. N. Frederickson and J. Ja.Ja, "Approximation algorithms for several graph augmentation problems," *SIAM Journal of Computing*, vol. 10-2, pp. 270–283, 1981.

[15] D. Harel and R. E. Tarjan, "Fast algorithms for finding nearest common ancestors," *SIAM Journal on Computing*, vol. 13-2, pp. 338–355, 1984.

[16] H. N Gabow, Z. Galil, T. Spencer, and R. E. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Combinatorica*, vol. 6-2, pp. 109–122, 1986.