Traffic Engineering for Provisioning Restorable Hose-Model VPNs

Yu-Liang Liu[†], *Nonmember*, Yeali S. Sun[†a)] and Meng Chang Chen[††b)], *Member*

Summary
Virtual Private Networks (VPNs) are overlay networks established on top of a public network backbone with the goal of providing a service comparable to Private Networks (PNs). The recently proposed VPN *hose-model* provides customers with flexible and convenient ways to specify their bandwidth requirements. In order to meet specified bandwidth requirements, the Network Service Provider (NSP) must reserve a sufficient amount of bandwidth on the data transmission paths between each endpoint pair of a VPN. In addition, reliability of a VPN depends on the reliability of data transmission paths. Italiano et al. proposed an algorithm that finds a set of backup paths for a given VPN (*VPN tree*) under the single-link failure model [1]. When any link failure on the *VPN tree* is detected, a backup path corresponding to the failed link can be activated to restore the disconnected VPN tree into a new one, and hence can ensure reliability of the given VPN. However, Italiano's algorithm cannot guarantee the specified bandwidth requirement of the given VPN under the single-link failure model will be met. To address this issue a new backup path set selection algorithm called *BANGUAD* is proposed in this work. In addition, issues about establishing multiple bandwidth-guaranteed *hose-model* VPNs under the single-link failure model have not been investigated. In this paper, we propose a bandwidth sharing algorithm as well as three provisioning algorithms for establishing multiple bandwidth-guaranteed *hose-model* VPNs under the single-link failure model. Experimental simulations that compare the performance of the proposed algorithms are reported.

Keywords: ***Virtual Private Network, Hose model, Failure Restoration, Traffic Engineering***

1. Introduction

Traditionally, a private network is established by grouping dedicated lines connecting several geographically dispersed sites (endpoints). As the number of endpoints is growing, connecting them with dedicated lines is becoming increasingly expensive. As a result, VPNs have emerged as replacements for PNs in recent years. The VPN is an overlay network that is established on top of a public packet switched network backbone intended to provide a service comparable to a PN. The two most important issues that must be addressed for VPNs are data security and bandwidth guarantees. The former is usually achieved by cryptographic methods, while the latter is achieved by reserving sufficient bandwidths on the backbone links.

The two most common VPN resource-provisioning models are: (1) the *customer-pipe model* [2-4] and (2) the *hose model* [3, 4]. In the *customer-pipe model*, customers must have precise predictions about the complete traffic requirements of each endpoint pair in a VPN in advance. The NSP then finds a data transmission path, $p_{u,v}$, for traffic between each endpoint pair, (u,v), of a VPN and allocate sufficient bandwidth for the path according to the traffic requirements. However, customers may be unwilling to or unable to know the traffic requirements of each endpoint pair in a VPN. This is especially true when there are a large number of endpoints per VPN.

In the *hose model*, customers only need to specify the ingress bandwidth requirement, $b^-(v)$, and egress bandwidth requirement, $b^+(v)$, for each endpoint, v, of a VPN. The value $b^-(v)$ is the maximum rate of traffic that endpoint v receives from the network at any time, and the value $b^+(v)$ is the maximum rate of traffic that endpoint v sends into the network. We only consider *hose-model* VPNs in our work as the *hose-model* appears to provide customers with more flexibility and convenience in specifying their bandwidth requirements.

It is important to consider two kinds of failures, namely link failures and router failures. A common fault model for link failures assumed in literature and justified by network measurements [5, 6] is that at any given time only one link in the network fails. In other words, in the event of a link failure, no other link fails until the failed link is restored; and the probability of two or more links failing at the same time is very small. In our work, we use the single-link failure model to devise several bandwidth-guaranteed *hose-model* VPNs provisioning algorithms. Moreover, we hereafter term the restorable bandwidth-guaranteed *hose-model* VPNs under the single-link failure model as *restorable VPNs*.

In order to meet the bandwidth requirement specified by customers, the NSP needs to reserve in a *restorable VPN* a sufficient amount of bandwidth on data transmission paths (or paths, for short) between each endpoint pair. We termed *primary bandwidth* as the bandwidth needed on the paths under the non-failure case. The additional bandwidth needed on the alternative paths under the link failure case is called the *protected bandwidth*. The *restorable VPN* provisioning algorithms proposed in this paper can be implemented on an MPLS network, as the path pinning capacity provided by MPLS technology can be used to

[†]The authors are with the Department of Information Management, National Taiwan University, Taiwan.
[††]The author is with the Institute of Information Science, Academia Sinica, Taiwan.
a)E-mail: sunny@im.ntu.edu.tw
b)E-mail: mcc@iis.sinica.edu.tw

direct the routing of the paths in a VPN [9, 10].

We organized the remainder of this paper into six sections. Section 2 is a review of related works. Section 3 introduces the problem considered in this paper. In Section 4, we propose a new backup path set selection algorithm called *BANGUAD* to address the problem of Italiano's algorithm mentioned previously. Section 5 proposes a bandwidth sharing algorithm as well as three *restorable VPN* provisioning algorithms. We also report experimental simulations that compare the performance of the proposed *restorable VPN* provisioning algorithms and Section 6 concludes this paper.

2. Related Works

Duffield et al. first introduced the VPN *hose-model* in [3, 4]. In their papers, they also proposed several *hose-model* VPN provisioning algorithms under the non-failure case. Kumar et al. argued that the provisioning of *hose-model* VPN should be based on a tree topology (hereafter called: *VPN tree*) and proposed the tree routing algorithm [7]. Given the bandwidth requirements of each endpoint in a VPN, the tree routing algorithm tries to find the *VPN tree* which needs minimum total bandwidth allocation on tree links (hereafter called: *bandwidth-optimization VPN tree*). In the case of symmetric bandwidth requirements (i.e., $b^+(v) = b^-(v)$ for all VPN endpoints v), the tree routing algorithm is optimal and is guaranteed to find the *bandwidth-optimization VPN tree*. However, In the case of asymmetric bandwidth requirement, the problem of finding the *bandwidth-optimization VPN tree* has proven to be *NP-hard*, and the tree routing algorithm is a 10-approximation algorithm.

Jűttner et al. compared the bandwidth allocation efficiency of the *hose-model* VPN with that of the *customer-pipe model* VPN [8]. Gupta et al. investigated the issues concerning MPLS labels design and routing protocol on a *VPN tree* [10]. Italiano et al. proposed a backup paths set selection algorithm for a *VPN tree* [1]. The goal of Italiano's work was to find a set of backup paths for the given *VPN tree* such that it could be restored under any single-link failure. Italiano's algorithm tries to find the backup path set that needs the minimum total *protected bandwidth* allocation. However, this problem is *NP-complete*, and the Italiano's algorithm is a 16-approximation algorithm. We also addressed this problem in our previous work, in which we proposed a new backup path set selection algorithm called *GBPH*. The *GBPH* needs less *protected bandwidth* allocation than Italiano's algorithm [11]. Chou proposed a multi-objective traffic-engineering framework for off-line provisioning of multiple *customer-pipe model* VPNs [12]. The goal of Chou's framework was to minimize the maximum link utilization on the network backbone while minimizing the total bandwidth allocation for establishing VPNs. Liu et al. point out that the *hose-model* VPNs provisioning algorithms proposed in [3, 4, 7] cannot achieve a satisfactory *rejection ratio* when multiple *hose-model* VPNs are to be established on-line on an MPLS network backbone. They proposed a new provisioning algorithm called the *Modified Tree Routing Algorithm* (*MTRA*) to address this issue [18].

3. Problem Formulation and Modeling

In this section, we formulate the problem considered in this paper. The MPLS network backbone managed by the NSP on which *restorable VPNs* are established is modeled in subsection 3.1. The *restorable VPN* is described in subsection 3.2. We model the VPN setup requests describing the *restorable VPNs* requested by customers in subsection 3.3. Finally, we describe the main problem considered in this paper (called *On-line Restorable VPNs Establishment Problem* (*ORVEP*)) in subsection 3.4.

3.1 Network Backbone Modeling

The MPLS network backbone is modeled by an undirected graph $G=(N,L)$, where N and L are the set of routers and the set of links, respectively. Let n and m denote the cardinality of N and L and B be the set of residual bandwidths on links of L. The amount of residual bandwidth on link l ($l \in L$) is denoted by $B(l)$. Each endpoint e_i in a VPN gains access to VPN service by connecting to a specific router r_i in N. We called router r_i the VPN access router of endpoint e_i. In other words, for each VPN endpoint, there is a corresponding access router in N.

3.2 The Restorable VPN

Due to the excellent bandwidth allocation efficiency of adopting tree topology in provisioning hose-model VPN [8], we assume that the paths $p_{u,v}$ between each endpoints pair (e_u,e_v) of a VPN follow a *VPN tree* in this paper. Let *vt* and *vt'* denote the *VPN trees* that connect all the access routers used by the *restorable VPN* under the non-failure and link failure cases, respectively. In order to meet the specified bandwidth requirement under the no-failure case, sufficient amount of *primary bandwidth* allocation is needed on links of *vt*. In addition, to ensure reliability of the paths in the VPN under the single-link failure model, a set of backup paths, $BP=\{bp_1,bp_2,\ldots bp_k\}$ corresponding to *vt* must be built. When any faulty link, *e*, on *vt* is detected, a backup path, $bp_e \in BP$, that corresponds to *e* can be activated to recover the disconnected *VPN tree* into a new *VPN tree*, *vt'* (i.e., $vt'=vt-e+bp_e$). Then the new path between each endpoints pair (e_u,e_v) of the VPN follow the path in the new *VPN tree vt'* (until *e* is repaired). However, in order to guarantee that the specified bandwidth requirement can

also be met when any tree link failure occurs, the NSP may needs to reserve a certain amount of additional bandwidth on links of *vt* and on links of the backup paths in *BP*. We term *protected bandwidth* as the additional bandwidth needed in this case.

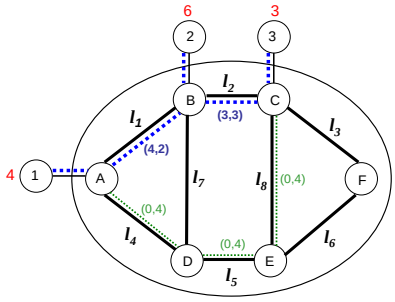


Fig. 1. An example of a restorable VPN

An illustration of a *restorable VPN* is shown in Fig. 1. The number besides endpoints e_1, e_2 and e_3 represents their bandwidth requirement. The thick dotted lines form a *VPN tree vt* consisting of links l_1 and l_2. The thin dotted lines form a backup path bp_1, which is used to recover any failure of tree links l_1 and l_2. In this example the backup path set *BP* only consisting one backup path, bp_1. The two numbers in parenthesis besides the links of *vt* and links of bp_1 represent the *primary bandwidth* and *protected bandwidth* needed on them. For example, the *primary bandwidth* and *protected bandwidth* needed on the tree link l_1 are 4 and 2, respectively. Under the non-failure case, the 4 units of *primary bandwidth* allocation on l_1 are enough to meet the specified bandwidth requirement. However when the l_2 fails, the backup path bp_1 is activated, and forms a new VPN tree *vt'* (i.e., $vt'=vt-l_2+bp_1$), which consisting of tree links l_1, l_4, l_5 and l_8. The maximum traffic rate through l_1 of *vt'* is 6, and 2 units of *protected bandwidth* allocation is needed. Given the bandwidth requirement of each endpoint on a *VPN tree*, for the rule of determining the maximum traffic rate through a tree link, please refer to [7]. Note that the italiano's algorithm do not allocate protected bandwidth on links of *vt* (l_1 and l_2 in this example), and hence cannot guarantee to meet the specified bandwidth requirement of the VPN

3.3 VPN Setup Requests Modeling

In this paper, we consider that the bandwidth requirement of each endpoint e_j is symmetric (i.e., $b^+(v) = b^-(v)$ for all VPN endpoints v), and use $b(e_j)$ to denote the bandwidth requirement of e_j. Each VPN setup request describes a *restorable VPN* from the customer for the NSP to establish, and let vr_i be the ith VPN setup request. Each vr_i is represented by a n-tuple vector $(h_1,h_2,\ldots,h_n)$, where n is the cardinality of the set N. The number of nonzero elements in vr_i represents the number of endpoints contained in the corresponding *restorable VPN*. The value of jth element, h_j, of vr_i represents the bandwidth requirement of endpoint e_j. *Maxr* denotes the maximum bandwidth guarantee provided by the NSP.

3.4 On-line Restorable VPNs Establishment Problem

The *ORVEP* defined in this paper is similar to the work in [13-16] which mainly considers on-line establishment of restorable bandwidth-guaranteed point-to-point paths. However, in the context of VPN provisioning, the basic unit of concern is a *restorable VPN* consisting of numerous restorable point-to-point paths (as opposed to one restorable point-to-point path) that makes the problem more challenging.

In *ORVEP*, the NSP manages an MPLS network backbone *G* (as described in section 3.1) on which *restorable VPNs* are established. We also assume that a link failure may occur on *G* at any given time. We consider the situation where (a) VPN setup requests arrive one-by-one independently, and (b) the NSP does not have a priori knowledge about future VPN setup requests. This knowledge includes the number of future VPN setup requests, the number of endpoints contained in each VPN setup request, and the bandwidth requirement of each endpoint. In this situation, the NSP must process each VPN setup request in an on-line manner.

Upon receiving a VPN setup request vr_i, the NSP triggers the *restorable VPN* provisioning algorithm to establish a corresponding VPN. The *restorable VPN* provisioning algorithm performs this task by first choosing a VPN tree *vt* and a backup paths set *BP*. The *restorable VPN* provisioning algorithm then allocates bandwidth on links of *vt* and links of the backup paths in *BP* (as described in section 3.2). If there is not enough residual bandwidth on the link when the bandwidth is being allocated, vr_i will be rejected.

In *ORVEP*, we use the *amount of bandwidth allocation* and the *rejection ratio* as the performance metrics to compare different VPN provisioning algorithms. The optimization goal is to minimize the *amount of bandwidth allocation* and the *rejection ratio* for establishing on-line *restorable VPNs*. The *rejection ratio* is defined as:

$$\text{rejection ratio} = \frac{\text{number of requests rejected}}{\text{total numbers of requests received}}$$

Minimizing the *amount of bandwidth allocation* to establish restorable VPNs maximizes the amount of bandwidth left over for other QoS-guaranteed traffic to coexist on the MPLS network backbone. Minimizing the *rejection ratio* maximizes the number of requests successfully established on the network backbone, and hence maximizes the service revenue of the NSP

In the *ORTVEP*, we assume that the NSP uses a server-based strategy [17] for processing VPN setup requests. In a server-based strategy, the *restorable VPN* provisioning algorithm runs on a single entity called *VPN request server* (*VRS*). The *VRS* also keeps the complete link state topology database and is responsible for computing an

explicit path for each endpoint pair of a *restorable VPN*. The paths then can be setup using a signaling protocol such as RSVP or CR-LDP. For computing the explicit paths, the *VRS* needs to know the current network topology and link residual bandwidth. We assume that a link-state routing protocol exists for information acquisition.

4. Bandwidth-Guaranteed Backup Paths Set Algorithm

In this section, we propose a new backup path section algorithm called *BANwidth-GUAranteeD backup paths Set Algorithm* (*BANGUAD*). Before explaining the algorithm in more detail, we need to define some notation. Table 1 summarizes the notation used in the algorithm. The pseudo code of *BANGUAD* is described in Table 2.

Table1. The notation used in *BANGUAD*

Symbol	Description
H=(N,L-$L(vt)$)	A network graph obtained by removing links on vt from G
$N(vt)$	Routers set on the input VPN tree vt
$L(vt)$	Links set on the input VPN tree vt
$sp_{u,v}$	A shortest path from router u to router v in the graph H, where $u,v \in N(vt)$
$L(BP)$	Link set which comprise links used by $bp_1 \cup bp_2 \ldots \cup bp_k$, where bp_i is a backup path in the output set BP
$cbp(l)$	The vector which indicate the corresponding backup path for the link l, where $l \in L(vt)$
$COVER(p)$	A subset of links in $L(vt)$, where insert a path p into vt will create a cycle containing all of them
$cover(p)$	$\{l \in COVER(p) \mid cbp(l) = \emptyset\}$
$sp_{u,v}$	A shortest path from node u to node v in graph H, where u and v are nodes in $N(vt)$
SP	A candidate backup path set which consisting the shortest paths $sp_{u,v}$ for each (u,v) pair, where u and v are nodes in $N(vt)$
$PB(l)$	The amount of *primary bandwidth* allocation on link l for vr_i, where $l \in L(vt)$
$pb(l)$	The amount of *protected bandwidth* allocation on link l for vr_i, where $l \in (L(vt) \cup L(BP))$
$\alpha(p)$	Total additional amount of *protected bandwidth* required on all link l ($l \in COVER(p) \cup p$), if p is included into BP in this iteration

In *BANGUAD*, the backup path set *BP* is first found out (from line 6 to line 10), then the amount of *protected bandwidth* allocation needed on links l ($l \in L(vt) \cup L(BP)$) is computed (from line 12 to line 18). Initially, the set *BP*, *SP* and the corresponding backup path for each tree link are set to null value (Ø) in line 1 and line 2. Because the links of all backup paths in *BP* must be link-disjoint with those of *vt* (i.e., $L(vt) \cap L(BP) = \emptyset$), the *BANGUAD* search candidate backup paths on the network graph *H*. The code from line 3 to line 5 compute a shortest path $sp_{u,v}$ for each tree router pair (u,v) and include them as elements of the candidate backup path set *SP*. Note that if there is no paths between the tree router pair (u,v), the value of $sp_{u,v}$ is set to null value.

Table2. The pseudo code of *BANGUAD*

Bandwidth-Guaranteed backup paths Set Algorithm	
Input: 1. A VPN setup request vr_i, 2. a network graph $G=(N,L)$, 3. A *VPN tree*, vt that connects all VPN access routers used in vr_i and 4. The amount of *primary bandwidth* allocation $PB(l)$ on links $l \in L(vt)$ for vr_i.	
Output: 1. A set of backup paths, $BP=\{bp_1, bp_2, \ldots bp_k\}$ for vt such that vt can tolerate any single link failure, 2. A vector which indicates the corresponding backup path for the link l, where $l \in L(vt)$and 3. The *protected bandwidth* allocation $pb(l)$ on all links $l \in L(vt) \cup L(BP)$ for vr_i.	
Algorithm:	
1.	$BP := \emptyset$; $SP := \emptyset$;
2.	For each link $l \in L(vt)$ {$cbp(l) := \emptyset$;}
3.	For each distinct node pair (u,v) $(u,v \in N(vt))$
4.	{ $sp_{u,v} := Compute\_Shortest\_Path(u,v)$;
5.	if ($sp_{u,v} \neq \emptyset$) $SP := SP \cup sp_{u,v}$; }
6.	Repeat { $p := Select\_Minimum\_Cost\_Path(SP)$;
7.	For each $l \in cover(p)$ { $cbp(l) := p$; }
8.	$BP := BP \cup p$; $SP := SP - p$;
9.	For each $p \in SP$ {$Update\_cost(p)$;}
10.	} Until ($cbp(l) \neq \emptyset$, for all $l \in L(vt)$);
11.	For eack link $l \in L$ { $pb(l) := 0$}
12.	For each link $e \in L(vt)$
13.	{ $vt' := vt - e + cbp(e)$;
14.	for each $l \in L(vt')$ {
15.	$max\_traffic(l) := Compute\_Max\_Traffic(vt', vr_i)$;
16.	if ($max\_traffic(l) > (PB(l) + pb(l))$)
17.	{$pb(l) := max\_traffic(l) - PB(l)$;} }
18.	}
19.	Output (BP);
20.	For each $l \in (L(vt) \cup L(BP))$ {Output($pb(l)$);}

The code from line 6 to line 10 is a loop which picks some paths from the candidate backup paths set *SP* to form the output set *BP*. The loop iterates until all the links on the input *VPN tree* find out their corresponding backup path (i.e., $cbp(l) \neq \emptyset$, for all $l \in L(vt)$ as described in line 10). The cost function associated to each path p in the set *SP* is defined as:

$$Cost(p) = \frac{\alpha(p)}{|cover(p)|}$$

In the loop from line 6 to line 10, a minimum cost path p in the set *SP* is first selected in line 6. Then p is assigned as the corresponding backup path to all the tree links in $cover(p)$ in line 7. The path p is added to the set *BP* and removed from the set *SP* in line 8. As the assignment of the corresponding backup path to tree links in $cover(p)$ may changes the cost values associated with some candidate backup paths in *SP* (i.e., the denominator of the cost function associated with some paths may change), the cost value associated with each path in *SP* is updated accordingly in line 9.

The amount of protected bandwidth allocation on link l, $pb(l)$, is determined by the difference between the maximum traffic through l under any possible tree link failure and the amount of *primary bandwidth* allocation on l. The code from line 12 to line 18 computes the $pb(l)$ value for all links l on $(L(vt) \cup L(BP))$. Initially, the amount *protected bandwidth* allocation $pb(l)$ on all links l is set to zero in line 11. The for statement in line 12 considers the case of any tree link e ($e \in L(vt)$) fail. When any tree link e fail, it's corresponding backup path $cbp(e)$ is activated and form a new VPN tree vt' in line 13. Given the new *VPN tree* vt' and a setup request vr_i, the code in line 15 computes the maximum traffic rate through l, where l are links on vt'. If the maximum traffic rate through l exceeds the total allocated bandwidth (i.e., $max\_traffic(l) > PB(l)+pb(l)$), the protected bandwidth allocation $pb(l)$ is updated in line 17.

We use fig. 2 to explain the *BANGUAD*. The input VPN setup request vr_1=(1, 5, 3, 2, 0, 0) and the input *VPN tree* is shown as the solid dotted lines. The input *VPN tree* consist links l_1, l_2 and l_7, with 1, 3 and 2 units of *primary bandwidth* allocation, respectively. Initially, the corresponding backup path of tree links l_1, l_2 and l_7 are set to null value and the *protected bandwidth* allocations on all links are set to 0. Before the first execution of the code from line 6 to line 10, the candidate backup paths set $SP=\{sp_{A,D}, sp_{C,D}\}$. The shortest path $sp_{A,D}$ is composed of link l_4 and the shortest path $sp_{C,D}$ are composed of l_8 and l_5. The cost value associated to candidate backup path $sp_{A,D}$ and $sp_{C,D}$ are 2.5 and 5.5. And hence, after the first execution of the code from line 6 to line 10, the $sp_{A,D}$ is selected. Because the $cover(sp_{A,D})=\{l_1, l_7\}$, so the corresponding backup path for tree links l_1 and l_7 are set to $sp_{A,D}$.

In the second execution of the code from line 6 to line 10, the $sp_{C,D}$ is selected. Because the $cover(sp_{C,D})=\{l_2\}$, so the corresponding backup path for tree links l_2 is set to $sp_{C,D}$. The *BANGUAD* output the set *BP* which consists of two backup paths $sp_{A,D}$, $sp_{C,D}$ shown as the thin dotted lines. After the execution of the code from line 11 to line 20, the amount *protected bandwidth* allocation on link l ($l \in L(vt) \cup L(BP)$) is shown in the second number in the parenthesis.

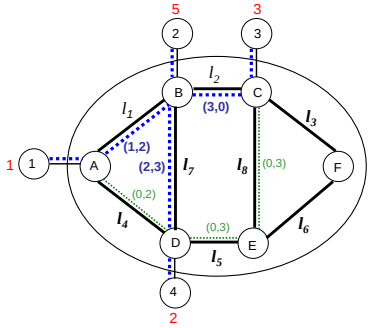


Fig. 2. An illustration f the BANGUAD

5. Algorithms under ORVEP

In this section, we propose algorithms under *ORTVEP*. In subsection 5.1, we elaborate the concept of bandwidth sharing between multiple restorable VPN. In subsection 5.2, we propose a bandwidth-sharing algorithm that can be subsumed in *restorable VPN* provisioning algorithms to improve their performance. In subsection 5.3, we introduce three *restorable VPN* provisioning algorithms. In subsection 5.4, we present experimental simulations that compare the performance of the three provisioning algorithms.

5.1 Bandwidth Sharing in Restorable VPNs

When establishing a restorable bandwidth-guaranteed point-to-point path under the single-link failure model, both an active path and an alternative link-disjoint path are needed. Two restorable bandwidth-guaranteed point-to-point paths cannot share their allocated bandwidth on links of alternative paths if their active paths are not link-disjoint [13]. However, in the case of VPN provisioning under the single-link failure model, two *restorable VPNs* whose VPN trees are not link-disjoint may have potential in sharing their *protected bandwidth* allocation. We use an example to elaborate our argument.

When the NSP (or *VRS*) receive two VPN setup requests vr_1=(4,6,2,0,0,0) and vr_2=(0,2,3,0,0,4). In the non-sharing case, after establishing the two *restorable VPNs*, the sketch of G is shown in fig. 3. The endpoint $e_{i,j}$ represents the jth endpoint of the VPN, vr_i. Thick dotted lines and thick dashed lines depict the VPN trees corresponding to vr_1 and vr_2; thin dotted lines and thin dashed lines depict the backup paths corresponding to vr1 and vr_2, respectively. The numbers inside the parenthesis are the amount of *primary bandwidth* and *protected bandwidth* allocated on the links for vr_1 and vr_2. Note that the two VPNs overlap on l_2 and l_5.

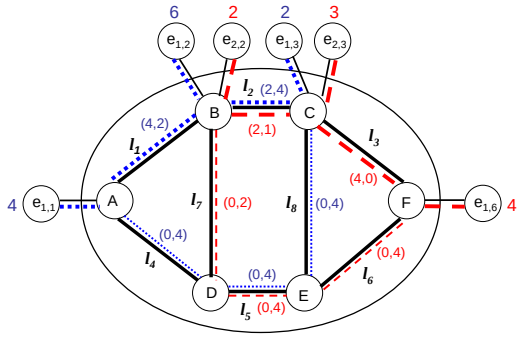


Fig. 3. The sketch of G after establishing two restorable VPNs

If the provisioning algorithm does not consider the sharing of bandwidth allocation among VPNs, the bandwidth on the links may be over-allocated. For example,

totally 9 and 8 units of bandwidth allocation are needed on l_2 and l_5, respectively to accommodate the two *restorable VPNs* in the non-sharing case (i.e., a total of 4 units of *primary bandwidth* and 5 units of *protected bandwidth* on l_2, and a total of 8 units of *protected bandwidth* on l_5). In this example, if the *restorable VPN* provisioning algorithm considers sharing bandwidth allocation among VPNs (after admitting and allocating bandwidth to the *restorable VPN* corresponding to vr_1), then there is no need for any additional *protected bandwidth* allocation on l_2 and l_5 when admitting the *restorable VPN* corresponding to vr_2. In subsection 5.2, we will show that if the bandwidth-sharing algorithm is subsumed to the *restorable VPN* provisioning algorithm, totally only 8 and 4 units of bandwidth allocation is enough on l_2 and l_5, respectively, to accommodate both the two *restorable VPNs*.

5.2 Bandwidth Sharing Algorithm for Restorable VPNs

To implement the bandwidth-sharing algorithm, the *VRS* must keep the amount of *primary bandwidth* and *protected bandwidth* that has been allocated, denoted by $PB_G(l)$ and $pb_G(l)$, respectively, for each link l on G. In addition, a traffic transfer matrix for G, denoted by $TTM(G)$ is also required. The value of $TTM_{i,j}(G)$ keeps the maximum amount of VPN traffic going through link l_i when link l_j fail.

When a VPN setup request vr_i is received, *VRS* computes the VPN flow transfer matrix $VFM(vr_i)$ associated with it. The value of $VFM_{i,j}(vr_i)$ keeps the maximum amount of traffic in the VPN of vr_i going through link l_i when the tree link l_j fails. Note that given the VPN tree vt and the backup path set BP for vr_i, the $VFM(vr_i)$ cam be computed easily. Let m be the cardinality of the set L, then both $VFM(vr_i)$ and $TTM(G)$ are m by m matrix. The bandwidth-sharing algorithm for restorable VPNs is described in table 3.

For ease of explanation, we assume that vr_i is always admitted. The amount of *primary bandwidth* allocation after admitting the vr_i is updated in line 1. Then The VPN flow transfer matrix associated with vr_i is computed in line 2. The traffic transfer matrix for G is updated from line 3 to line 5. The ith element of array *Max_row_traffic* keeps the maximum amount of traffic passing through link i under the single link failure case. In each iteration of the code from line 6 to line 12 the value of $Max\_row\_traffic[l_i]$ is first computed, then the value of $pb_G(l_i)$ for link l_i can be determined. The rule for determining the value of $pb_G(l_i)$ is that if the amount of bandwidth that has been allocated on link i (i.e., $PB_G(i)+ pb_G(i)$) is less than the value of $Max\_row\_traffic[i]$, then the value of $pb_G(i)$ is updated to the value of $(Max\_row\_traffic[i]-PB_G(i))$.

Table 3. The bandwidth-sharing algorithm for restorable VPNs

Bandwidth Sharing Algorithm for Restorable VPNs
Input: 1. The amount of *primary bandwidth* that has been allocated, denoted by $PB_G(l)$, and the amount of *protected bandwidth* that has been allocated, denoted by $pb_G(l)$, for each link l on G, 2. The traffic transfer matrix $TTM(G)$, 3. A VPN setup request vr_k, 4. The VPN tree vt and the backup path set BP for vr_i and 5 The *primary bandwidth* allocation $PB(l)$ on the link l of vt for vr_k.
Output: The amount of *primary bandwidth* that has been allocated, denoted by $PB_G(l)$ and the amount of *protected bandwidth* that has been allocated, denoted by $pb_G(l)$ on link l ($l \in L$) after admitting the request vr_k.
Algorithm:
1. for (each $l \in L(vt)$) $\{PB_G(l):= PB_G(l)+PB(l);\}$ 2. $VFM(vr_i):=Compute\_VFM(vr_i,vt,BP)$; 3. for ($i$=1 to m) { 4. for (j=1 to m) 5. $\{TTM_{i,j}(G):= TTM_{i,j}(G)+ VFM_{i,j}(vr_i);\}$ } 6. for (i=1 to m) { 7. $Max\_row\_traffic[l_i]:=0$; 8. for (j=1 to m) { 9. if ($TTM_{i,j}(G)>Max\_row\_traffic[l_i]$) 10. { $Max\_row\_traffic[l_i]:= TTM_{i,j}(G)$;} } 11. if ($Max\_row\_traffic[l_i]> (PB_G(l_i)+ pb_G(l_i))$) 12. { $pb_G(l_i):= Max\_row\_traffic[l_i]-PB_G(l_i)$;} }

We use the example in section 5.1 to illustrate the bandwidth-sharing algorithm we proposed. After processing vr_1=(4,6,2,0,0,0) and the execution of the code from line 1 to line 10 for vr_2=(0,2,3,0,0,4), the amount of *primary bandwidth* that has been allocated on links of L and the traffic transfer matrix for G (after the execution of the code from line 1 to line 13) are:

	$PB_G(l_i)$	$pb_G(l_i)$
l_1	4	2
l_2	4	4
l_3	4	0
l_4	0	4
l_5	0	4
l_6	0	0
l_7	0	0
l_8	0	4

$$TTM(G)=\begin{bmatrix} 0&6&0&0&0&0&0&0\\ 6&0&3&0&0&0&0&0\\ 0&3&0&0&0&0&0&0\\ 4&2&0&0&0&0&0&0\\ 4&4&4&0&0&0&0&0\\ 0&2&4&0&0&0&0&0\\ 0&2&4&0&0&0&0&0\\ 4&2&0&0&0&0&0&0 \end{bmatrix}$$

Note that from the above $TTM(G)$, we can obtain $Max\_row\_traffic[l_2]$=6 and $Max\_row\_traffic[l_5]$=4. After the execution of the code line 11 and line 12, there are no need for additional *protected bandwidth* allocation on links l_2 and l_5 (e.g., $Max\_row\_traffic[l_i]<=(PB_G(l_i)+ pb_G(l_i)$, for l_2 and l_5).

	$Max\_row\_traffic$
l_1	6
l_2	[6]
l_3	3
l_4	4
l_5	[4]
l_6	4
l_7	4
l_8	4

5.3 The Proposed Restorable VPN Provisioning Algorithms

A *restorable VPN* provisioning algorithm may contain the following three main components: (1) VPN provisioning algorithm for the non-failure case, (2) backup path set selection algorithm, and (3) resource-sharing mechanism. With the trade-off between the implementation complexity and performance benefit in mind, the NSP may design *restorable VPN* provisioning algorithms with different flavors by adopting various approaches for the three components. In this section, we proposed three provisioning algorithms for establishing on-line of *restorable VPNs* under *ORVEP*.

Algorithm A: Optimal-Tree without bandwidth sharing
- Find a VPN tree *vt* by using *tree routing* algorithm [7].
- Find the corresponding backup path set for *vt* by using *BANGUAD*.
- Disable the bandwidth-sharing algorithm introduced in section 5.2.

Algorithm B: Optimal-Tree with bandwidth sharing
- Same as Algorithm A, except the enabling of the bandwidth sharing-algorithm introduced in section 5.2.

Algorithm C: Enumerate-tree with bandwidth sharing
- For each node *v* on *G*, find a VPN tree vt_v by using breadth first search algorithm [18].
- For each VPN tree vt_v, find its corresponding backup path set BP_v by using *BANGUAD*.
- Select the combination of VPN tree and backup path set with minimum additional bandwidth allocation.
- Enable the bandwidth sharing algorithm introduced in section 5.2.

5.4 Experimental Simulations

To evaluate and compare the performance of the *restorable VPN* provisioning algorithms proposed in section 5.3, we have conducted two simulations. Due to extensive adaptation of the *KL topology* as the MPLS network backbone in the literature about MPLS traffic engineering [11, 13, 15, 18], we also adopt it as *G* in both simulations. Note that the *KL topology* is a network graph consisting of 15 routers and 28 links.

Simulation 1: The bandwidth allocation efficiency in the three restorable VPN provisioning algorithms

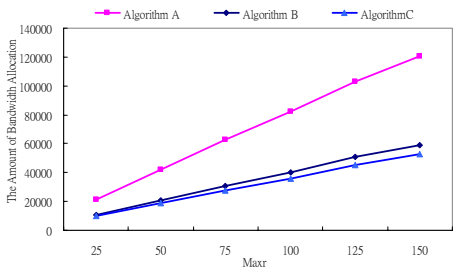
The parameter configuration of Simulation 1 is shown in Table 4. In this simulation, we consider the case where the MPLS network backbone *G* has a sufficient amount of residual bandwidth to accommodate all the restorable VPNs (i.e., all VPN setup requests received are accepted). Let *K* denote the total number of requests received. We use the total amount of bandwidth allocation for processing *K* restorable VPNs as the performance metric to compare different VPN provisioning algorithms.

Table 4. Parameter configuration of Simulation 1

$B(l_i)$	*p*	*Maxr*	*K*
∞	5	25~150 step 25	100

The simulation results are shown in fig. 4. The x-axis represents the value of *Maxr*, and the y-axis represents the amount of bandwidth allocation in the *restorable VPN* provisioning algorithms. The value of *Maxr* varies from 25 to 150 with a step of 25. We conducted 10 runs for each value of *Maxr*, and took the average amount of bandwidth allocation in these 10 runs. As expected, the average amount of bandwidth allocation increases as the value of *Maxr* increases in all three algorithms. In all the *Maxr* values considered in this simulation, the difference of the average amount of bandwidth allocation between *Algorithm A* and *Algorithm B* range from 50.25% to 51.69%. This means that the inclusion of the bandwidth-sharing algorithm into the VPN provisioning algorithms can reduce about half of the total bandwidth allocation in the provisioning algorithms. On the other hand, in all the *Maxr* values considered in this paper, the difference of the average amount of bandwidth allocation between *Algorithm B* and *Algorithm C* range from 8.17% to 11.20%. The bandwidth saving of *Algorithm C* over *Algorithm B* is caused by selection of a good combination of VPN tree and backup path set for all VPN setup requests.

Fig. 4. Comparison of the amount of bandwidth allocation in the three provisioning algorithms



Simulation 2: The rejection ratios achieved in the three provisioning algorithms

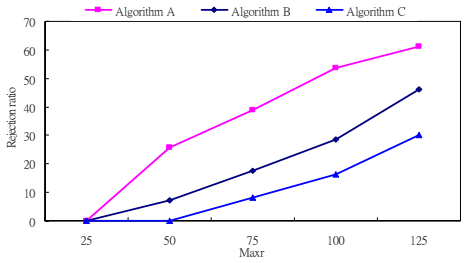
The parameter configuration of Simulation 2 is shown in Table 5. In this simulation, we consider the case where the MPLS network backbone *G* may not have sufficient amount of residual bandwidth to accommodate some VPNs (i.e., some VPN setup request received may be rejected). We use the rejection ratio for processing *K* VPN setup requests as the performance metric to compare *restorable VPN* provisioning algorithms.

Table 5. Parameter configuration of Simulation 2

$B(l_i)$	*p*	*Maxr*	*K*
Light links=3,000 units Dark links=12,000 units	5	25~150 step 25	100

The simulation results are shown in fig. 5. The x-axis represents the value of *Maxr*, and the y-axis represents the rejection ratios in the three *restorable VPN* provisioning algorithms. The value of *Maxr* varies from 25 to 150 with a step of 25. We conducted 10 runs for each value of *Maxr*, and took the average rejection ratios in these 10 runs. As expected, the average rejection ratios raise as the value of *Maxr* increases in all three algorithms. The average rejection ratio achieved by *Algorithm C* is much less than the other two provisioning algorithms in almost all the *Maxr* values considered in this simulation (except for the light load case, when the value of *Maxr* is 25, the average rejection ratios is 0% in all the three algorithms). On the other hand, the difference of the rejection ratios achieved by *Algorithm A* and *Algorithm B* is notable, meaning that the inclusion of the bandwidth sharing algorithm into the *restorable VPN* provisioning algorithms is very effective in reducing the rejection ratio.

Fig. 5. The comparison of rejection ratios achieved in the three VPN provisioning algorithms



6. Conclusions

Previous literatures about restoration issue in *hose-model* VPN focus on finding a backup path set for a given *VPN tree* such that the VPN can be restored under the single-link failure model [1,11]. However, the backup path selection algorithm proposed in these literatures can't guarantee to meet the specified bandwidth requirements under single-link failure model. To address this issue we proposed a new backup path selection algorithm for a VPN tree called *BANGUAD*. To our knowledge until now, issues about establishing multiple bandwidth-guaranteed *hose-model* VPNs on a MPLS network backbone under the single-link failure model have not been investigated. In this paper, we proposed a bandwidth sharing algorithm as well as three *restorable VPN* provisioning algorithms for on-line establishment of multiple bandwidth-guaranteed *hose-model* VPNs.

References

[1] G. Italiano, R. Rastogi and B. Yener, Restoration Algorithms for Virtual Private Networks in the Hose Model, in: *Proc. of IEEE INFOCOM*, 2002.

[2] B.S. Davie, Y. Rekhter, MPLS Technology and Applications, Morgan Kaufmann, San Francisco, CA, 2000.

[3] N. G. Duffield, P. Goyal and A. Greenberg, A Flexible Model for Resource Management in Virtual Private Networks, in: *Proc. of ACM SIGCOMM*, 1999.

[4] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan and J. E. V. D. Merwe, Resource Management with Hoses: Point-to-Cloud Services for Virtual Private Networks, IEEE/ACM Transactions on Networking 10(5) (2002) 679-692.

[5] T.H. Wu, Fiber Network Service Survivability, Artech House, Norwood, MA, 1992.

[6] D. Zhou, S. Subramaniam, Survivability in optical networks, IEEE Network 14 (6) (2000) 16 – 23.

[7] A. Kumar, R. Rastogi, A. Silberschatz and B. Yener, Algorithms for Provisioning Virtual Private Networks in the Hose Model, IEEE/ACM Transactions on Networking 10(4) (2002) 565-578.

[8] A. Jűttner, I. Szabo and Á Szentesi, On Bandwidth Efficiency of the Hose Resource Management Model in Virtual Private Networks, in: *Proc. of IEEE INFOCOM*, 2003.

[9] D. O. Awduche, j. Malcom, J. Agobua, M. O'Dell and J. Mcmanus, Requirement for Traffic Engineering over MPLS, IETF RFC 2702, September 1999.

[10] A. Gupta, A. Kumar and R. Rastogi, Exploring the Trade-off between Label Size and Stack Depth in MPLS Routing, in: *Proc. of IEEE INFOCOM*, 2003.

[11] Y. L. Liu, Y. S. Sun, M. C. Chen, Traffic Engineering for Hose-Model VPN Provisioning, in: *Proc. of IEEE GLOBECOM*, 2005.

[12] C. T. Chou, Traffic Engineering for MPLS-based Virtual Private Networks, Computer Networks 44(3) (2004) 319 – 333.

[13] M. Kodialam and T. T. Lakshman, Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration, in: *Proc. of IEEE INFOCOM*, 2000.

[14] M. Kodialam and T. T. Lakshman, Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information, in: *Proc. of IEEE INFOCOM*, 2001.

[15] S. Raza, F. Aslam and Z. A. Uzmi, Online Routing of Bandwidth Guaranteed Paths with Local Restoration using Optimized Aggregate Usage Information in: *Proc. of IEEE International Conference on Communications*, 2005.

[16] L. E. Li, M. M. Buddhikot, C. Chekuri and K. Guo, Routing Bandwidth Guaranteed Paths with Local Restoration in Label Switched Networks in: *Proc. of IEEE International Conference on Network Protocols*, 2002.

[17] G. Apostolopoulos, R. Guérin, S. Kamat, S. K. Tripathi, Server-Based QoS Routing, in: *Proc. of IEEE GLOBECOM*, 1999.

[18] Y. L. Liu, Y. S. Sun, M. C. Chen, MTRA: An On-Line Hose-Model VPN Provisioning Algorithm, Springer Journal on Telecommunication Systems, to appear. Available online: http://www.iis.sinica.edu.tw/LIB/TechReport/tr2004/tr04020.pdf.