Telecommun Syst (2006) 31:379–398
DOI 10.1007/s11235-006-6724-2

MTRA: An on-line hose-model VPN provisioning algorithm

Yu-Liang Liu · Yeali S. Sun · Meng Chang Chen

Abstract Virtual private networks (VPNs) provide customers with a secure and manageable communication environment. The allocation of bandwidth for VPNs to meet the requirements specified by customers is now one of the most important research issues in the field of traffic engineering. A VPN resource-provisioning model called *hose-model* was developed to provide customers with a flexible and convenient way to specify the bandwidth requirements of a VPN. Several *hose-model* VPN provisioning algorithms have already been proposed. They focus on the bandwidth efficiency issue in the case of establishing a single *hose-mode* VPN. However, these algorithms cannot achieve a satisfactory *rejection ratio* when: (1) the residual bandwidths on links of the network backbone are finite and (2) multiple VPN setup requests are handled on-line. In this paper, we propose a new *hose-model* VPN provisioning algorithm called *MTRA* to address the issue. *MTRA* can process multiple VPN setup requests rapidly and reduce the *rejection ratio* effectively. Theoretical upper bounds of *rejection ratios* achieved by several VPN provisioning algorithms are also derived. The experiments verify that *MTRA* performs better in regards to the *rejection ratio* than other provisioning algorithms.

Keywords Virtual private network · Hose-model · VPN provisioning algorithms · Traffic engineering

1. Introduction

Traditionally, a private network (PN) 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 [1]. As a result, VPNs

Y.-L. Liu · Y. S. Sun
Department of Information Management, National Taiwan University, Taipei, Taiwan
e-mail: {d8725001; sunny}@im.ntu.edu.tw

M. C. Chen
Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: mcc@iis.sinica.edu.tw

have emerged as replacements for PNs in recent years. The VPN is a logical network that is established on top of a packet switched network backbone. Its goal is to provide a service comparable to a PN. The two most important issues that must be addressed for VPN 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 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 in advance about the complete traffic requirements of each endpoint pair in a VPN. The Network Service Provider (NSP) then finds a data transmission path, $path_{u,v}$, for traffic between each endpoint pair (u, v), in a VPN and allocate sufficient bandwidth for the path according to the traffic requirement. However, customers may be unwilling, or unable, to know the traffic requirements of each endpoint pair in a VPN. This is especially true when the number of endpoints per VPN is large.

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. As the *hose-model* appears to provide customers with more flexibility and convenience in specifying their bandwidth requirements, we only consider *hose-model* VPNs in this paper.

The most important VPN provisioning algorithms for *hose-model* VPNs are: (1) *provider-pipes* [3, 4], (2) *hose-specific state* [3, 4], (3) *VPN-specific state* [3, 4], and (4) *tree routing* [5]. For the approaches of selecting a data transmission path, $path_{u,v}$, between each endpoint pair, (u,v), in a VPN and the allocated bandwidth on links of the paths in these algorithms, please refer to [3, 5, 6]. The path pinning capacity provided by MPLS (multiprotocol label switching) technology can be used to direct the routing of a data transmission path between each endpoint pair in a VPN [7, 8]. Our approach can be implemented on a MPLS network as well.

VPN provisioning algorithms can be implemented in two ways: (1) off-line provisioning and (2) on-line provisioning. In off-line provisioning, the NSP has a prior knowledge of all VPN setup requests. In this setting, the VPN provisioning plan is optimized on some performance metrics (e.g., revenue, network link utilization and the amount of bandwidth reservation) by rejecting selected requests. In the on-line provisioning, when a VPN setup request is received, it is processed based on the current state of the network. As the NSP does not know future VPN setup requests, the on-line decision only achieves optimal provisioning for the current network state. In this paper, we focus on on-line VPN provisioning.

To our knowledge until now, issues about the *rejection ratios* achieved by *hose-model* VPN provisioning algorithms have not been investigated. In this paper, we consider the problem of minimizing the *rejection ratio* of provisioning algorithms when (1) the residual bandwidths on links of the network backbone are finite, and (2) multiple VPNs need to be established on-line on the network backbone. Once the data transmission paths between each endpoint pair in a VPN are determined, the provisioning algorithm needs to explicitly allocate sufficient bandwidth on the links of these paths to meet the bandwidth requirement specified by customers. As the bandwidth allocation of VPNs is executed on-line, the previous allocation may affect the feasibility of the next VPN provisioning. One of the requisites of a good VPN provisioning algorithm is that it should achieve a low *rejection ratio*. However, previous *hose-model* VPN provisioning algorithms [3–5] have been unable to meet this requirement. We therefore propose a new provisioning algorithm, the *Modified Tree Routing Algorithm* (*MTRA*), to address this issue. Our experimental simulations show that the *MTRA* can reduce the *rejection ratio* effectively. In addition, it can also rapidly process multiple VPN setup requests. Given a network graph G with n nodes and m edges, *MTRA* spends only O(mn) time for a VPN setup request.

The contributions of this paper are summarized as follows: (1) We show by concrete examples that all four of the *hose-model* VPN provisioning algorithms mentioned previously are unable to achieve satisfactory *rejection ratios*. To address this issue, we propose a new *hose-model* VPN provisioning algorithm called *MTRA*. (2) The theoretical upper bounds of the *rejection ratios* achieved by the *provider-pipes*, *tree routing* and *MTRA* provisioning algorithms for the problem we consider are also derived in this paper.

The remainder of this paper is organized as follows. In Section 2, we review related works. In Section 3, we define the *On-line Hose-model VPN Establishment Problem* (*OHVEP*) where an NSP establishes *hose-model* VPNs online on a network backbone composed of links with finite residual bandwidths. In *OHVEP*, the performance metric for comparison with various VPN provisioning algorithms is the *rejection ratio*. In Section 4, we exemplify the reasons why the provisioning algorithms proposed in [3–5] cannot achieve a satisfactory *rejection ratio*. In Section 5, we present *MTRA*. In Section 6, we derive the theoretical upper bounds of the *rejection ratios* for several *hose-model* VPN provisioning algorithms under the *OHVEP*. In Section 7, we show five experimental simulations to compare the performance of *MTRA* with other VPN provisioning algorithms. Finally, in Section 8, we give our conclusions and indicate the direction of our future work.

2. Related works

The *hose-model* was first introduced by Duffield et al. in [3, 4]. In their papers, *provider-pipes, hose-specific state* and *VPN-specific state* provisioning algorithms for *hose-model* VPNs were also presented. Duffield et al.'s work inspired other researchers to develop provisioning algorithms for *bandwidth-optimization hose-model VPNs*. Kumar et al. argued that *bandwidth-optimization hose-model VPNs* should be based on a tree topology (hereafter called: *VPN tree*) [5]. They also presented an algorithm to compute the *VPN tree* which needs minimum total bandwidth allocation on tree links (hereafter called: *bandwidth-optimization VPN tree*) where the links on the network backbone have infinite capacity and the bandwidth requirement of each endpoint is symmetric (i.e., $b^+(v) = b^-(v)$ for all VPN endpoint $\boldsymbol{v}$). If the links on the network backbone have infinite capacity and the bandwidth requirement of each endpoint is general, Kumar et al. proved that it is *NP-hard* to compute the *bandwidth-optimization VPN tree* and proposed a 10-approximation algorithm to solve the problem. Gupta et al. improved the approximation ratio to 9.002 [9]. Swamy and Kumar further reduced the ratio to 5 [10]. In the case where the links on the network backbone have finite capacity, Gupta et al. also proved that computing the *bandwidth-optimization VPN tree* is *NP-hard* [9]. Note that *NP-hard* is a class of problems with tremendous computational complexity. For more details of *NP-hard,* please refer to [11].

Juttner et al. compared the bandwidth allocation efficiency of the *hose-model* VPN with that of the *customer-pipe model* VPN [6]. They also conducted simulations to compare the bandwidth allocation efficiency of the four *hose-model* VPNs provisioning algorithms mentioned in Section 1. Italiano et al. proposed a restoration algorithm for a *hose-model VPN tree* under the single-link failure model [1]. Balasubramanlan and Sasaki compared the bandwidth allocation efficiency of several restoration algorithms for a *hose-mode VPN tree* under the single-link failure model through experimental simulations [12]. Gupta et al. investigated the issues about MPLS labels design and routing protocol for a *VPN tree* [8]. Chou proposed a multi-objective traffic-engineering framework for off-line provisioning of a series of *M customer-pipe model* VPNs [13]. The goal of Chou's framework is to minimize the maximum link utilization on the network backbone while minimize the total bandwidth allocation for establishing the *M* VPNs.

3. Problem formulation and modeling

In this section, we formulate the problem considered in this paper. The network backbone managed by the NSP is modeled in subsection 3.1. The VPN setup request describing the VPN service requested by customers is modeled in subsection 3.2. Finally, the *On-line Hose-model VPNs Establishment Problem (OHVEP)* is described in subsection 3.3.

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, respectively. Let B be the set of residual bandwidths of links on L, and the amount of residual bandwidth on link $l (l \in L)$ is denoted by $B(l)$. A subset $AR = \{ar_1, ar_2, \ldots, ar_p\}$ of $N (AR \subseteq N)$ is the set of VPN access routers. Each endpoint e_i of a VPN gains access to VPN service by connecting to a specific VPN access router ar_i in AR. In other words, for each endpoint of a VPN, there is a corresponding VPN access router in AR.

The elliptic region in Fig. 1 is an example of the MPLS network backbone G. The round regions (A to G) inside G are routers in N. The solid lines between any two routers in L are links in L. The number beside each link is the amount of residual bandwidth on it ($B(l) = 5$ for all $l \in L$ in this figure). The VPN access routers set $AR = \{A, E, G\}$. The round regions (1, 2 and 3) outside G are endpoints (e_1, e_2 and e_3, respectively, in our notation) of a VPN which gain access to VPN service via routers in AR. The dotted lines labeled as $path_{i,j}$ is the data transmission path for VPN traffic between e_i and e_j.

3.2. VPN setup request modeling

The demands for VPN service of customers are described by VPN setup requests. In this paper, we consider that the bandwidth requirement of each endpoint e_j is symmetric and use $b(e_j)$ to denote the bandwidth requirement of e_j. Let $Maxr$ denote the maximum bandwidth guarantee provided by the NSP, and vr_i be the ith VPN setup request from customer for the NSP to establish. Each vr_i is represented by a p-tuple vector $(r_1, r_2, \ldots, r_p)$, where p is the cardinality of the access routers set AR. The number of nonzero elements in vr_i represents the number of endpoints contained in the corresponding VPN. The value of jth element, r_j, of vr_i represents the bandwidth requirement of endpoint e_j.

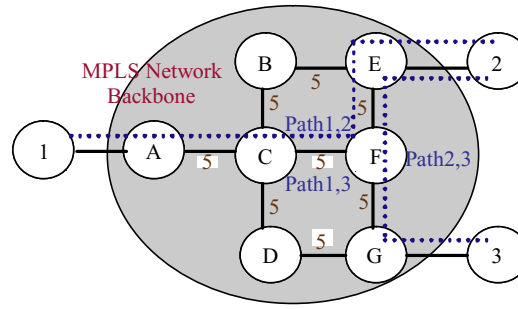


Fig. 1 An example of MPLS network backbone G

3.3. On-line hose-model VPNs establishment problem

The *OHVEP* defined in this paper is similar to the work in [14–18] which mainly considers on-line establishment of bandwidth-guaranteed point-to-point tunnels. However, in the context of VPN provisioning, the basic unit of concern is a VPN consisting of numerous point-to-point tunnels, as opposed to one point-to-point tunnel, that makes the problem more challenging.

In *OHVEP*, the NSP manages an MPLS network backbone G (as described in subsection 3.1) on which VPNs are established. We consider the situation where (a) VPN setup requests arrive one-by-one independently, and (b) the NSP do 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 provisioning algorithm to establish a corresponding VPN. The provisioning algorithm performs this task by first choosing a data transmission path, $path_{u,v}$, between each endpoint pair (u, v), and then allocating bandwidth on each link of the path. If there is not enough residual bandwidth on the link when the bandwidth is being allocated, vr_i will be rejected. We use the *rejection ratio* as the performance metric to compare different *hose-model* VPN provisioning algorithms. Note that the authors of [14–18] also use the *rejection ratio* (of tunnel setup requests) as the performance metric to compare different on-line tunnel establishment algorithms. The *rejection ratio* is defined as:

$$\text{Rejection ratio} = \frac{\text{Number of requests rejected}}{\text{Total numbers of requests received}}$$

The optimization goal of provisioning algorithms is to minimize the *rejection ratio*, which in turn will maximize the number of requests successfully established on the network backbone.

In the *OHVEP*, we assume that the NSP uses a server-based strategy [19] for processing VPN setup requests. In a server-based strategy, the VPN provisioning algorithm is run 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 data transmission path for each endpoint pair of a VPN. Then the paths 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 for information acquisition exists.

4. Motivation for new provisioning algorithms

In subsection 4.1, we exemplify the reasons why the four provisioning algorithms proposed in [3–5] cannot achieve satisfactory *rejection ratios* under *OHVEP*. We present two scenarios to support our argument. Then, in subsection 4.2, we list the factors influencing the *rejection ratios* achieved by provisioning algorithms.

4.1. The drawbacks of other algorithms

Scenario 1: The higher bandwidth allocation of *provider-pipes, hose-specific state* and *VPN-specific state* results in higher *rejection ratio* than *tree routing*

Under the same routing pattern, the following relationship holds for the bandwidth allocated on each link between different provisioning algorithms to establish a VPN (the relation also

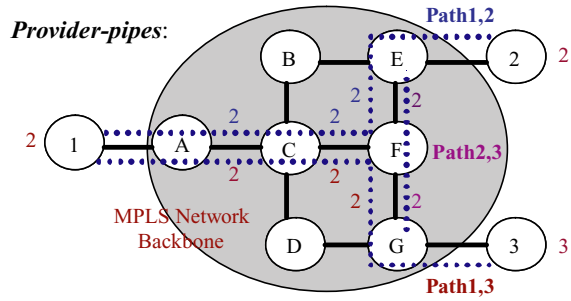


Fig. 2 A sketch of G for Scenario 1

holds for total bandwidth allocation):

$$BW_{Provider\text{-}pipes} \geq BW_{Hose\text{-}specific} \geq BW_{VPN\text{-}specific} \text{ [6]} \tag{6}$$

In addition, the simulation results in [6] show that the allocated bandwidth of *tree routing* is less than that of *VPN-specific*. To highlight the difference between the allocated bandwidths of the provisioning algorithms, we compare *provider-pipes* with *tree routing* in this scenario.

When the NSP receives a VPN setup request, $vr_1 = (2, 2, 3)$, in the case of adopting the *provider-pipes* algorithm, the resulting allocations on the backbone G are shown in Fig. 2. The numbers beside the three endpoints represent their bandwidth requirements ($b(e_1)$, $b(e_2)$, and $b(e_3)$). The numbers beside the dotted lines represent the amount of bandwidth needed on the respective links. Note that the amount of allocated bandwidth on $l_{A,C}$, $l_{C,F}$, $l_{E,F}$ and $l_{F,G}$ is 4 in the *provider-pipes* algorithm, whereas it is 2, 2, 2, and 3, respectively in the *tree routing* algorithm. Moreover, the *provider-pipes* algorithm has over-allocated bandwidth on $l_{A,C}$, $l_{C,F}$, $l_{E,F}$ and $l_{F,G}$. For example, the traffic rate through $l_{A,C}$ at any instant will not exceed $min\,(b(e_1), b(e_2) + b(e_3))$, which is equal to 2. However, the *provider-pipes* algorithm has allocated 4 units of bandwidth to it (a similar problem also occurs on $l_{C,F}$, $l_{E,F}$ and $l_{F,G}$).

Scenario 1 illustrates the difference between the allocated bandwidths of different provisioning algorithms in establishing a single VPN. In the case of establishing multiple VPNs, the difference between the allocated bandwidths of the *provider-pipes, hose-specific state*, and *VPN-specific state* algorithms (compared with the *tree routing* algorithm) will be greater. If the residual bandwidths on links in L are finite, the phenomenon will result in a higher *rejection ratio* in *provider-pipes, hose-specific state*, and *VPN-specific*. In Scenario 2, we show that even the *tree routing algorithm* cannot guarantee a satisfactory *rejection ratio*.

Scenario 2: Disregarding the amount of residual bandwidths on links in *tree routing* algorithm results in a higher *rejection ratio*

When the NSP receives two VPN setup requests $vr_1 = (2, 3, 3)$ and $vr_2 = (3, 3, 3)$, the result is shown in Fig. 3. Initially, the residual bandwidth on all links in L is 5 units (the numbers beside the solid lines). The round region labeled as $e_{i,j}$ represents the jth endpoint of the VPN, vr_i. The number beside each $e_{i,j}$ represents its bandwidth requirement. In the case of the *tree routing* algorithm, the *VPN trees* corresponding to vr_1 and vr_2 are depicted as the trees formed by dotted lines and dashed lines, respectively. The numbers beside dotted lines and dashed lines represent the amount of bandwidth allocated on respective links. In this figure, neither $l_{E,F}$ nor $l_{F,G}$ have enough bandwidth to accommodate the second request after processing the first one. The *rejection ratio* achieved by the *tree routing* algorithm in Scenario 2 is 50%.

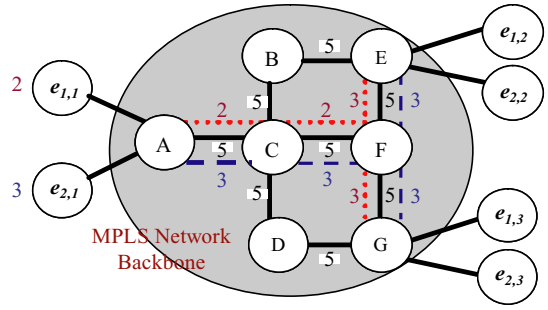


Fig. 3 A sketch of G for Scenario 2

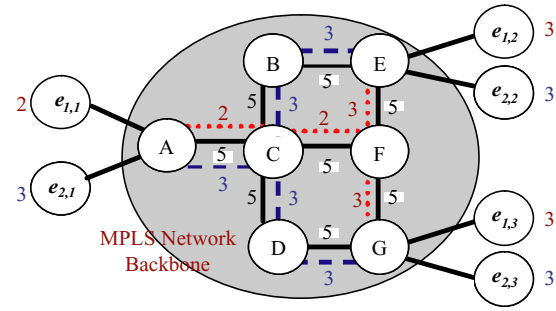


Fig. 4 Optimal arrangement for Scenario 2

In fact, the amount of available resources on G is enough to accommodate both requests. If we rearrange the *VPN tree* of vr_2 as shown by the dashed lines in Fig. 4, then both vr_1 and vr_2 can be accepted in this case. The *rejection ratio* achieved by this rearrangement is 0%.

The *tree routing* algorithm may still reject requests even though the amount of available resources on G is sufficient to process them. This is because the *tree routing* algorithm insists on using the links forming the *bandwidth-optimization VPN tree* for each request, regardless of the amount of residual bandwidths on them. If the amount of residual bandwidths on the links of the *bandwidth-optimization VPN tree* is thinly spread, it is obvious that the optimization behavior of *tree routing* will raise the likelihood of rejection.

4.2. The factors influencing rejection ratio

In this case, the links of the network backbone have a finite amount of residual bandwidth and the NSP needs to establish multiple VPNs on the network backbone on-line (as described in the *OHVEP*). The two most important factors influencing the *rejection ratio* achieved by the provisioning algorithms are:

(1) *Bandwidth allocation efficiency*: As mentioned in Scenario 1 of subsection 4.1, this issue has been widely discussed in previous literature [3–6].
(2) *A load balancing mechanism that considers the amount of residual bandwidth on links*: As described in Scenario 2 of Subsection 4.1, provisioning algorithms must take the residual bandwidths of links into account and avoid using links that are thinly spread. This will balance the load on G and reduce the *rejection ratio*.

5. MTRA

To alleviate the drawbacks of (a) inefficiency on bandwidth allocation, and (b) disregarding the amount of residual bandwidth for links selection (described in subsection 4.1), we propose a new provisioning algorithm called the *Modified Tree Routing Algorithm (MTRA)*. The *tree routing* and *MTRA* provisioning algorithms are both tree-based (i.e., they establish a VPN base on tree topology (*VPN tree*)). While *tree routing* has excellent bandwidth allocation efficiency, it does not consider maximizing the accommodation of on-line VPN requests. On the contrary, *MTRA* considers both bandwidth allocation efficiency and accommodation of on-line VPN requests by achieving balance of link residual bandwidths.

The major difference between *tree routing* and *MTRA* is that the cost function they defined for *VPN tree* selection. Let T be a *VPN tree* consisting of k links. The cost functions of *tree routing* and *MTRA* are defined as following:

$$Cost_{MTRA}(T) = \sum_{1 \le x \le k} \frac{RS(l_x)}{B(l_x)}, \quad \text{and } Cost_{treerouting}(T) = \sum_{1 \le x \le k} RS(l_x),$$

where $RS(l_x)$ and $B(l_x)$ represent the amount of bandwidth allocation needed and the amount of residual bandwidth on the xth link, l_x, respectively. The cost function of *MTRA* is derived by the cost function defined in the routing algorithms proposed in [17, 20] for route selection.

When processing a request, *MTRA* tries to find a *VPN tree* that minimizes the cost function defined above. It is clear the additional cost for using a link l_x in building a *VPN tree* is proportional to the value of $RS(l_x)$ and is reciprocal to the value of $B(l_x)$. Therefore, *MTRA* tries to find a *VPN tree* with links of abundant residual bandwidths and low overall bandwidth allocation. As a result,

Table 1 Pseudo code for *MTRA*

***Modified Tree Routing Algorithm* (*MTRA*)**

Input: A Network graph $G = (N, L)$, VPN access routers $AR = (ar_1, ar_2, \ldots, ar_p) \subseteq N$, residual bandwidth constraints B on L, and a VPN setup request $vr_i = (r_1, r_2, \ldots, r_p)$.
Output: A minimum cost VPN tree VT_{MC} for vr_i, on which all leaf nodes are VPN access routers ar_j with $r_j > 0$.
Algorithm:

```
1. VT_MC := Ø;
2. For each v∈N
3. {
4.    T_v := BFS_Tree(G, v);
5.    PT_v := Prune_Tree(T_v, vr_i);
6.    Compute_RS(PT_v, vr_i);
7.    if(Cost(PT_v) < Cost(VT_MC))VT_MC := PT_v;
8. }
9. if (Cost(VT_MC) = ∞)
10.   {Reject(vr_i); Return Ø;}
11. else {
12.    For each link l_x ∈ VT_MC{B(l_x) := B(l_x) − RS(l_x);}
13.    Accept(vr_i); Return(VT_MC);
14.   }
```

MTRA can satisfy both bandwidth allocation efficiency and balance of residual bandwidths. The pseudo code of *MTRA* is described in Table 1.

Given a network graph G consisting of n nodes, to process a VPN setup request vr_i, *MTRA* iterates totally n times, once for each $v \in N$. In each iteration, *MTRA* first finds a candidate *VPN tree* PT_v rooted at v for vr_i, and then computes the amount of bandwidth needed to be allocated to each link l_x of PT_v. Finally the cost value associated with PT_v can be computed. After finding all $PT_v(v \in N)$, if there is not any $PT_v(v \in N)$ on which all links have enough residual bandwidth for allocation, *MTRA* will reject vr_i. In the case of accepting vr_i, *MTRA* will return the *VPN tree* with the minimum cost value among all $PT_v(v \in N)$ for vr_i, which is denoted by VT_{MC}. In addition, *MTRA* then allocates bandwidth to each link l_x of VT_{MC} by performing $B(l_x) := B(l_x) - RS(l_x)$.

To find a candidate *VPN tree* PT_v rooted at v, *MTRA* first find a *BFS tree* (breadth first search tree [21]) T_v rooted at v (by calling Function *BFS_Tree*). T_v contains all nodes in G and in addition, T_v may contain nodes that are not VPN access routers used in vr_i as leaf nodes. Therefore, *MTRA* prunes T_v and obtains a candidate *VPN tree* PT_v, on which all leave nodes are VPN access routers used in vr_i (by calling Function *Prune_Tree*).

MTRA computes the amount of bandwidth needed for each link l_x of a *VPN tree* T according to the bandwidth requirement information in vr_i (by calling Function *Compute_RS* in Table 2). To compute the value of $RS(l_x)(l_x \in T)$, we first remove l_x from T which partitions the *VPN tree* into two subtrees T_x^a and T_x^b. Let $BR\_T_x^a$ and $BR\_T\ _x^b$ denote the accumulated bandwidth requirement of the VPN access routers (endpoints) on T_x^a and T_x^b, respectively. Then $RS(l_x)$ is determined by the minimum value of $BR\_T_x^a$ and $BR\_T_x^b$. For more details about computing the $RS(l_x)$ value for each l_x on a *VPN tree*, please refer to [5].

Given a *VPN tree* T, in a normal case, the function *Cost* of *MTRA* returns the cost value computed by the cost function defined previously. However, where T is null (Ø), or there are links on T that do not have enough bandwidth for allocation, the function Cost will return ∞.

The time complexity of each iteration in *MTRA* is O(m), which is determined by the function *BFS_Tree*. To process a request, a total of n iterations are required. So, It is clear that the time complexity of *MTRA* for processing a request is O(mn).

Table 2 Pseudo code for *Compute_RS*

Function *Compute_RS*(T, vr_i)

Let l_x be the xth link on T.
Let $RS(l_x)$ be the amount of bandwidth allocation needed on l_x with respect to the bandwidth requirement specified in vr_i.
Let T_x^a and T_x^b be the two subtrees obtained by remove l_x from T.
1. for (each element $r_j \neq 0$ $(1 \leq j \leq p)$ of vr_i)
2. {
3. Initialize two variable $BR\_T_x^a$, $BR\_T_x^b$ to value 0;
4. For (each element $r_j \neq 0$ $(1 \leq j \leq p)$ of vr_i)
5. {
6. if($ar_j \in T_x^a$) then add r_j to $BR\_T_x^a$
7. else add r_j to $BR\_T_x^b$
8. }
9. $RS(l_x)$:=*min*($BR\_T_x^a$, $BR\_T_x^b$);
10. }

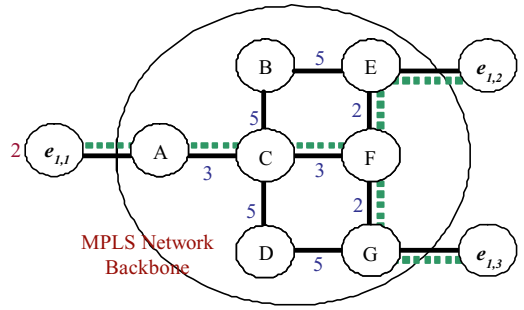


Fig. 5 A sketch of G after processing vr_1

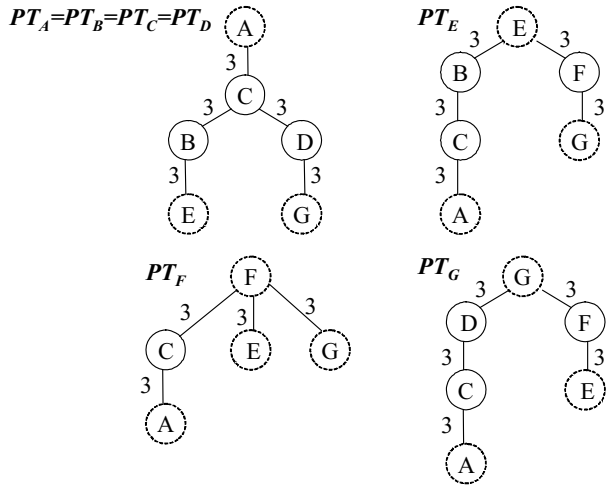


Fig. 6 Candidate *VPN trees* considered for vr_2

We now consider Scenario 2 in Section 4 and adopt *MTRA* to process requests. Initially, $B(l) = 5$ for all l in L. The sketch of G, after accepting vr_1, is shown in Fig. 5. The number beside each link in G is its residual bandwidth after accepting vr_1. The dotted lines form the minimum cost *VPN tree* VT_{MC} that *MTRA* will output for vr_1.

After accepting vr_1, *MTRA* then processes vr_2. Each candidate *VPN tree* $PT_v(v \in N)$ for vr_2 considered by *MTRA* is shown in Fig. 6. We can find four different types of candidate *VPN tree* for vr_2. Note that PT_A, PT_B, PT_C and PT_D are identical. The number beside each link of $PT_v(v \in N)$ is the amount of bandwidth that needs to be allocated to it. The cost value associated with each $PT_v(v \in N)$ is:

$$
\begin{aligned}
&Cost(PT_A) = Cost(PT_B) = Cost(PT_C) = Cost(PT_D) \\
&= \frac{RS(l_{A,C})}{B(l_{A,C})} + \frac{RS(l_{B,C})}{B(l_{B,C})} + \frac{RS(l_{C,D})}{B(l_{C,D})} + \frac{RS(l_{B,E})}{B(l_{B,E})} + \frac{RS(l_{D,G})}{B(l_{D,G})} \\
&= \frac{3}{3} + \frac{3}{5} + \frac{3}{5} + \frac{3}{5} + \frac{3}{5} = 3.4,
\end{aligned}
$$

and

$$Cost(PT_E) = Cost(PT_F) = Cost(PT_G) = \infty.$$

It is clear that *MTRA* will return PT_A for vr_2 (vr_2 is accepted by *MTRA*). Hence the *rejection ratio* achieved by *MTRA* in Scenario 2 is 0%.

6. Theoretical upper bounds of the rejection ratios

In this Section, we will derive theoretical upper bounds of the *rejection ratios* achieved by *provider-pipes, tree routing* and *MTRA* in the scenario of *OHVEP*. The NSP can evaluate the upper bounds of the *rejection ratios* achieved by provisioning algorithms before processing requests according to the parameter configuration in *OHVEP*. Let K denote the total number of requests received. Recall that the parameters in *OHVEP* are K, p, B and *Maxr*, where p is the number of VPN access routers on G, $B = (B(l_1), B(l_2), \ldots, B(l_m))$ is the residual bandwidth on links of L and *Maxr* is the maximum bandwidth guarantee provided by the NSP.

We define constants $B_{\min} = \text{Min}\{B(l_1), B(l_2), \ldots, B(l_m)\}$,$RS_{pp}^{\max} = \frac{p*(p-1)}{2} * Maxr$ and $RS_{\text{tree}}^{\max} = \lfloor \frac{p}{2} \rfloor * Maxr$. We also define an artificial request $vr_{\max} = (Maxr, Maxr, \ldots, Maxr)$. In this section, a new parameter q, which represents the number of link-disjoint candidate *VPN trees* $PT_v(v \in N)$ that *MTRA* can find for $vr_{\max}$ on G, was also introduced.

Definition 1. Given two vectors $a = (a_1, a_2, \ldots, a_p)$ and $b = (b_1, b_2, \ldots, b_p)$, a is defined to be not less than b, denoted by $a \geq b$, if all elements in $a - b$ are all non-negative.

Property 1. *Given a network graph $G = (N, L)$ on which the residual bandwidths on links of L are finite, two VPN setup requests $vr_a = (a_1, a_2, \ldots, a_p)$ and $vr_b = (b_1, b_2, \ldots, b_p)$ with $vr_a \geq vr_b$, if vr_a is accepted by a deterministic provisioning algorithm PA in G, then vr_b is also accepted by PA in G.*

Lemma 1. *Given an arbitrary network graph G with p VPN access routers, residual bandwidth constraint B on L, and a sequence of K one-by-one requests with the maximum bandwidth requirement of each endpoint no more than Maxr, then the rejection ratio in provider-pipes will not exceed*

$$\begin{cases} 0 & , \text{if } K \leq \left\lfloor \dfrac{B_{\min}}{RS_{pp}^{\max}} \right\rfloor \\ 1 - \dfrac{B_{\min} - RS_{pp}^{\max}}{K * RS_{pp}^{\max}} & , \text{if } K > \dfrac{B_{\min}}{RS_{pp}^{\max}} \end{cases}$$

Proof: To establish a VPN, the *provider-pipes algorithm* must construct a provider pipe $pp_{i,j}$ between each endpoints pair (e_i, e_j) and allocate bandwidth to it. A provider-pipe $pp_{i,j}$ is a path from e_i to e_j in G and the amount of bandwidth needed on each link of this path is $\min\{b(e_i), b(e_i)\}$. To establish a VPN containing p endpoints, the number of provider pipes needed to be constructed is $p^*(p-1)/2$.

Let K received requests be $vr_1, vr_2, \ldots, vr_K$ in sequence. We also produce another K artificial requests $vr_1', vr_2', \ldots, vr_K'$, where the value of each $vr_i'(1 \leq i \leq K)$ is equal to $vr_{\max}$.

First, we consider processing $vr_1', vr_2', \ldots, vr_K'$ with the *provider-pipes algorithm*. For each $vr_i'(1 \leq i \leq K)$ accepted by the *provider-pipes algorithm*, the amount of bandwidth needed on any link of G will not exceed $RS_{pp}^{\max}$. Therefore, all of $vr_1', vr_2', \ldots, {}^{vr}\lfloor \frac{B_{\min}}{RS_{pp}^{\max}} \rfloor'$ will be accepted by the *provider-pipes algorithm*. Because $vr_i' \geq vr_i(1 \leq i \leq K)$, according to *property*

$1, vr_1, vr_2, \ldots, {}^{vr}\lfloor \frac{B_{\min}}{RS^{\max}_{pp}} \rfloor$ will also be accepted by the *provider-pipes algorithm*. Thus the lemma follows. □

Lemma 2. *If the situation is the same as described in Lemma 1, the rejection ratio in any tree-based hose-model VPN provisioning algorithm will not exceed*

$$\begin{cases} 0 & , \text{if } K \le \left\lfloor \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \right\rfloor \\ 1 - \dfrac{B_{\min} - RS^{\max}_{\text{tree}}}{K * RS^{\max}_{\text{tree}}} & , \text{if } K > \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \end{cases}$$

Proof: Let K received requests be $vr_1, vr_2, \ldots, vr_K$ in sequence. We also produce another K artificial requests $vr'_1, vr'_2, \ldots, vr'_K$, where the value of each $v'_i (1 \le i \le K)$ is equal to $vr_{\max}$.

First, we consider processing $vr'_1, vr'_2, \ldots, vr'_K$ with a tree-based provisioning algorithm. For each $vr'_i (1 \le i \le K)$ accepted by the tree-based provisioning algorithm, the algorithm will find a VPN tree $vt'_i (1 \le i \le K)$ for it. The amount of bandwidth allocated on each link of $vt'_i (1 \le i \le K)$ for each accepted $vr'_i (1 \le i \le K)$ will not exceed $RS^{\max}_{\text{tree}}$. Therefore, all of $vr'_1, vr'_2, \ldots, vr'_{\lfloor \frac{B_{\min}}{RS^{\max}_{\text{tree}}} \rfloor}$ will be accepted by the tree-based provisioning algorithm. Because $vr'_i \ge vr_i (1 \le i \le K)$, according to Property 1, $vr_1, vr_2, \ldots, vr_{\lfloor \frac{B_{\min}}{RS^{\max}_{\text{tree}}} \rfloor}$ will also be accepted by the tree-based provisioning algorithm. Thus the lemma follows. □

Theorem 1. *If the situation is the same as described in Lemma 1, except that there exists $q (q \ge 2)$ link-disjoint VPN tree PT_v for VT_{max} on the given network graph* G*, the rejection ratio in MTRA will not exceed*

$$\begin{cases} 0 & , \text{if } K \le q * \left\lfloor \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \right\rfloor \\ 1 - \dfrac{q * (B_{\min} - RS^{\max}_{\text{tree}})}{K * RS^{\max}_{\text{tree}}} & , \text{if } K > q * \left\lfloor \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \right\rfloor \end{cases}$$

Proof: The proof is similar to that in Lemma 2, except that for each artificial request $vr'_i (1 \le i \le K)$, *MTRA* can find at least q PT_v for it. Note that *MTRA* will not reject a request, unless there do not exist any $PT_v (v \in N)$ on which all links have enough residual bandwidth for allocation. Therefore, all of $vr'_1, vr'_2, \ldots, vr_{q * \lfloor \frac{B_{\min}}{RS^{\max}_{\text{tree}}} \rfloor}$ will be accepted by *MTRA*. Thus the lemma follows. □

Corollary 1. *In the situation described as Theorem 1, the theoretical upper bound of the rejection ratio in the tree routing algorithm is still*

$$\begin{cases} 0 & , \text{if } K \le \left\lfloor \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \right\rfloor \\ 1 - \dfrac{B_{\min} - RS^{\max}_{\text{tree}}}{K * RS^{\max}_{\text{tree}}} & , \text{if } K > \dfrac{B_{\min}}{RS^{\max}_{\text{tree}}} \end{cases}$$

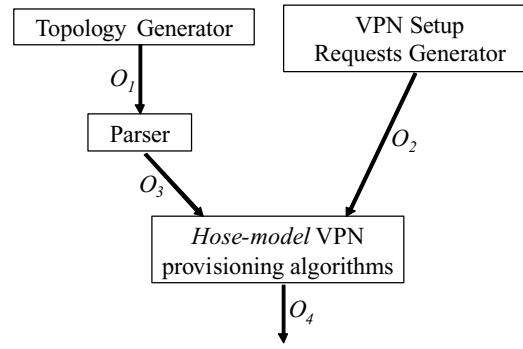


Fig. 7 The architecture of *HVPAS*

Proof: The proof is also similar to that in Lemma 2. Although for each artificial request $vr_i'(1 \leq i \leq K)$, the *tree routing* algorithm can find at least qPT_v for it. However, in the worst case, only one of them is the *bandwidth-optimization VPN tree* for $vr_i'(1 \leq i \leq K)$. The *tree routing* algorithm will insist on using this tree. Thus, the corollary follows. □

If all other conditions hold, increasing the value of p for the *provider-pipes* algorithm will raise the upper bounds of the *rejection ratio* in a speed of square order (see Lemma 1). However, for any tree-based *hose-model* VPN provisioning algorithms (*MTRA* and *tree routing*), increasing the value of p only raises the upper bounds of the *rejection ratio* in a speed of linear order (see Lemma 2). On the other hand, increasing the value of *Maxr* will raise the upper bounds of the *rejection ratio* of the three provisioning algorithms in a speed of linear order (see Lemmas 1 and 2). The *rejection ratio* upper bound of *MTRA* is superior to that of *tree routing* approximately q times (see Theorem 1 and Corollary 1). However, given a network graph G, the value of q depends on the density of G (the ratio of the number of links over the number of nodes in G (i.e., m/n)) and the distribution of VPN access routers on G. We investigate the effect of p, *Maxr*, and the density of G on the *rejection ratios* achieved by various provisioning algorithms in Section 7.

7. Simulation and performance results

7.1. Simulation environment

To evaluate the performance of *MTRA*, we set up a *hose-model* VPN provisioning algorithms simulator (*HVPAS*). The architecture of *HVPAS*, shown in Fig. 7, contains 4 main elements: (1) topology generator, (2) parser, (3) *hose-model* VPN provisioning algorithms, and (4) VPN setup requests generator. We implemented all components, except topology generator, in Java programming language.

The topology generator of *HVPAS* randomly generates the MPLS network backbone G administrated by the NSP. Because Brite [22, 23] has been widely used in a lot of research literature to generate random network topologies, we also adopt it as the topology generator in *HVPAS*. We generate randomly a connected network graph *by* assigning proper values in the configuration file used by Brite. The G output from the topology generator is parsed by the parser into a format readable by the provisioning algorithms of *HVPAS*. We have implemented three provisioning algorithms in *HVPAS*: (1) *MTRA*, (2) *tree routing*, and (3) *WSP provider-pipes*. The *WSP provider-pipes* algorithm is the same as the *provider-pipes* provisioning algorithm introduced in

Table 3 Parameter configuration of Simulation 1

G	$B(l_i)$	p	$Maxr$	K
KL topology	Light links = 1500 units Dark links = 6000 units	7	75	100

[3, 4]. However, the approach for selecting a path for each provider-pipe between endpoint pairs follows the *Widest Shortest Path* (*WSP*) algorithm [24]. The *WSP* is used to reduce the likelihood that there is not enough bandwidth on the chosen shortest path between endpoint pairs. The VPN setup requests generator in *HVPAS* randomly generates a set of VPN setup requests according to the given parameters K, p, and $Maxr$. The request set contains K requests. The number of endpoints contained in each VPN is generated randomly between 2 and p, and the bandwidth requirement $b(e_i)$ for each endpoint e_i is generated randomly between 1 and $Maxr$.

In Fig. 7, O_1 denotes G output by Brite in a specific format, O_3 denotes G in a format readable by provisioning algorithms implemented in *HVPAS*, and O_2 denotes requests generated by the VPN setup requests generator. Both O_2 and O_3 are input to the provisioning algorithms implemented in *HVPAS*. For each request accepted by a provisioning algorithm, O_4 represents a corresponding VPN topology. (Recall that for *MTRA* and *tree routing*, the VPN topology is a tree in G.)

7.2. Performance results

In this subsection, we describe five simulations. The first four compare the *rejection ratio* achieved by various provisioning algorithms implemented in *HVPAS*. The last simulation investigates the bandwidth allocation efficiency of *MTRA*.

Simulation 1: Performance comparison in KL *topology*

The parameter configuration of Simulation 1 is shown in Table 3. Due to extensive adaptation of the *KL topology* as the MPLS network backbone in the literature about MPLS traffic engineering [14–18], we also adopt it as G. The *KL topology* is composed of 15 routers and 28 links, as shown in Fig. 8. The routers labeled as ar_1 – ar_7 are VPN access routers, the amount of residual bandwidth on the light links is 1500 units, and the amount of residual bandwidth on the dark links is 6000 units.

We conducted 15 runs in Simulation 1, in which each run randomly generated 100 requests. The simulation results are shown in Fig. 9. The x-axis represents the run number and the y-axis represents the *rejection ratio* achieved by each provisioning algorithm in each run. We can see that the *rejection ratio* achieved by *MTRA* is much less than that achieved by *WSP provider-pipes* and *tree routing*. The *rejection ratios* achieved by *MTRA* are 0% in all runs except in run 8 and run 10 (where they are only 2 and 1%, respectively). However, the *rejection ratio*s by *WSP provider-pipes* and *tree routing* range from 30 to 55%. According to the simulation results, we believe that *MTRA* can reduce the *rejection ratio* effectively in the *KL topology*.

Simulation 2: The effect of *Maxr*

The parameter configuration of Simulation 2 is shown in Table 4. In order to evaluate the performance of *MTRA* on general G, we used Brite to randomly generate a connected

Fig. 8 The *KL Topology*

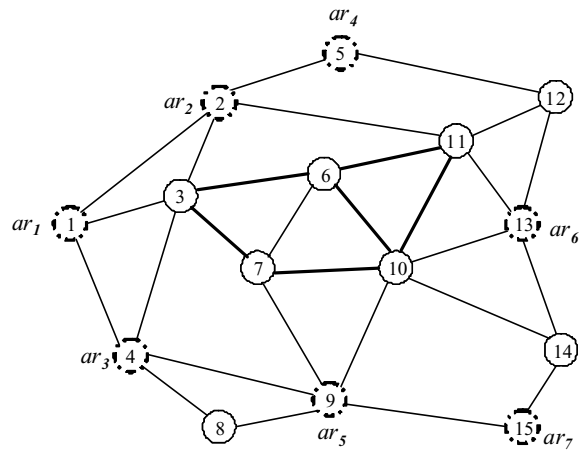
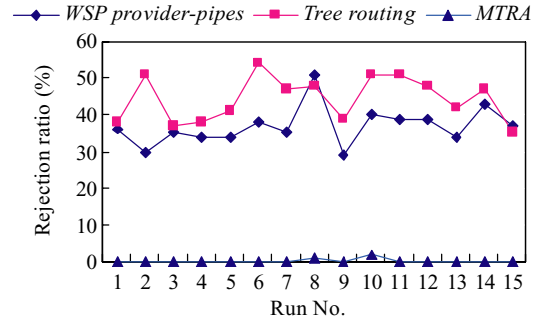


Fig. 9 Performance comparison in *KL Topology*



graph G with 20 nodes and 40 links in each run. The value of *Maxr* varies from 40 to 120 with a step of 20. We conducted 8 runs for each value of *Maxr*, and took the average *rejection ratio* achieved in these 8 runs.

The simulation results are shown in Fig. 10. The x-axis represents the value of *Maxr*, and the it y-axis represents the average *rejection ratio* achieved by the provisioning algorithms. As expected, the average *rejection ratio* increases as the value of *Maxr* increases in all three algorithms. The average *rejection ratio* achieved by *MTRA* is much less than the other two algorithms in almost all the *Maxr* values considered in this simulation (except for the light load case, when *Maxr*= 40, the average *rejection ratios* is 0% in all the three algorithms). The experimental results show that *MTRA* can indeed achieve a lower *rejection ratio* on general G compared to the other two algorithms.

Table 4 Parameter configuration of Simulation 2

G	$B(l_i)$	p	*Maxr*	K
Randomly generated by Brite with 20 nodes and 40 links	1500 units	6	40 – 120 step 20	100

Fig. 10 The effect of *Maxr*

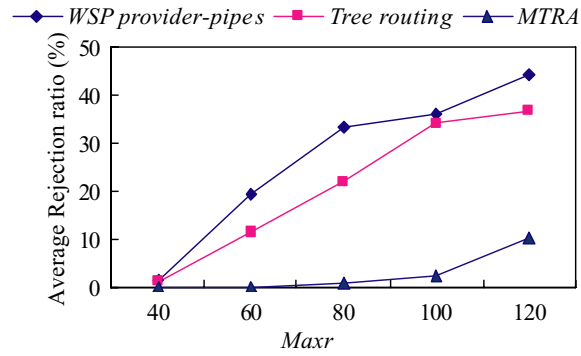


Table 5 Parameters configuration of Simulation 3

G	$B(l_i)$	p	$Maxr$	K
Randomly generated by Brite with 20 nodes and $\alpha(G) = 2-6$	1500 units	6	100	100

Simulation 3: The effect of $\alpha(G)$

The parameter configuration of Simulation 3 is shown in Table 5. We denote the ratio of the number of links over the number of nodes in G as $\alpha(G)$ (i.e., $\alpha(G) = m/n$). As we want to investigate the impact of $\alpha(G)$ on the average *rejection ratio*, we fix the value of n and change the value of G in this simulation. If all other conditions hold, increasing the value of $\alpha(G)$ indicates that: (1) the resources available for establishing VPN also increases and (2) the value of the parameter q may also increase. We conduct 8 runs for each value of $\alpha(G)$, and took the average *rejection ratio* achieved in these 8 runs.

The simulation results are shown in Fig. 11. The x-axis represents the value of $\alpha(G)$, and the y-axis represents the average *rejection ratio* achieved by the provisioning algorithms. As expected, in all three algorithms, the average *rejection ratio* declines as the value of $\alpha(G)$ increases. For all the $\alpha(G)$ values we consider in this experiment, *MTRA* achieved the lowest average *rejection ratio* among the three algorithms. On the other hand, as the value of $\alpha(G)$ increases, the reduction speed of the average *rejection ratio* in *tree routing* is slower than

Fig. 11 The effect of $\alpha(G)$ value

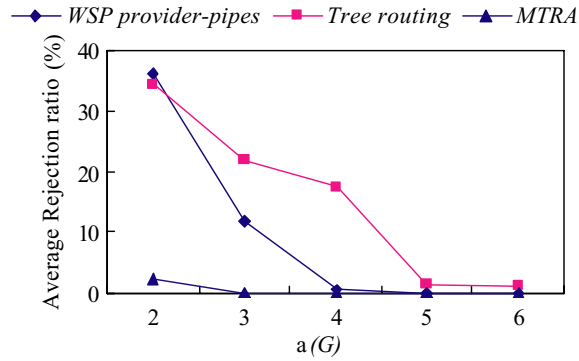


Table 6 Parameter configuration of Simulation 4

G	$B(l_i)$	p	$Maxr$	K
Randomly generated by Brite with 20 nodes and 40 links	1500 units	3 – 8	100	100

that of the *WSP-provider-pipes* algorithm. The reason is that *tree routing* insists on choosing a *bandwidth-optimization VPN tree* for each request, regardless of the amount of residual bandwidth on the links of the *VPN tree*. If the amount of residual bandwidth on any link of the *VPN tree* is insufficient for allocation, the request will be rejected. Therefore, the effect of an increase in $\alpha(G)$ is smaller in the *tree routing* than the *WSP provider-pipes* algorithm.

Simulation 4: The effect of p

The parameter configuration of Simulation 4 is shown in Table 6. This experiment investigates the impact of p on the average *rejection ratio*. The value of p varies from 3 to 8. For each p value, we conducted 8 experiments and took the average *rejection ratio* achieved in these 8 runs. If all other conditions hold, increasing the value of p has the following effects:

(1) The average load on links becomes heavy because the average number of endpoints contained in a request will increase. This effect will increase the *rejection ratio*.
(2) The fixed load is shared by more VPN access routers because we generate a fixed number of requests (totally K requests) in each run. Hence the fixed load is shared by more VPN access routers (more links on G). This effect will reduce the *rejection ratio*.

The simulation results are shown in Fig. 12. The x-axis represents the value of p, and the y-axis represents the average *rejection ratio* achieved by the provisioning algorithms. Of all the p values we consider in this experiment, *MTRA* achieves the lowest *rejection ratio* among the three algorithms. Both *MTRA* and *tree routing* have a transition point in the figure (i.e., when $p = 4$ for *MTRA* and when $p = 5$ for *tree routing*). Before the transition point, the effect of (1) is smaller than (2), and vice versa. However, for the *WSP provider-pipes* algorithm, as the value of p increases, the average *rejection ratio* rapidly rises. As the value of p increases, the additional bandwidth allocation increases at a speed of square order. (Recall that to establish a VPN containing p endpoints, the number of *provider-pipes* that needs to be constructed is $p^*(p-1)/2$.)

Fig. 12 The effect of p value

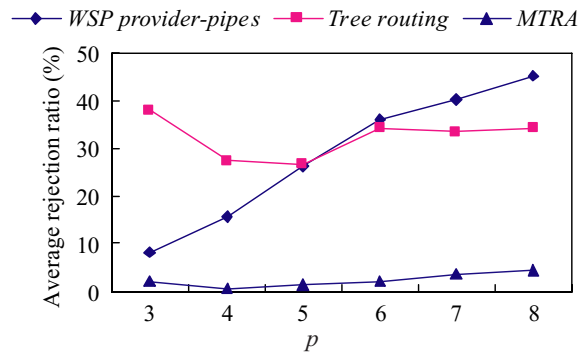


Table 7 Parameter configuration of Simulation 5

G	$B(l_x)$	p	$Maxr$	K
Randomly generated by Brite with 20 nodes and 40 links	5000 – 10000	6	100	100

Table 8 The average amount of allocated bandwidth for 100 requests

$B(l_x)$	RS_{MTRA}	$RS_{Tree\,routing}$	$Percent_{Extra\_BW}$
5000	21704	21138.38	2.6758%
6000	22184.63	21858.38	1.4926%
7000	21957.25	21721.13	1.0871%
7500	20377.63	20294.5	0.4096%
10000	21927.63	21872.5	0.252%

Simulation 5: The bandwidth allocation efficiency of *MTRA*

The parameter configuration of Simulation 5 is shown in Table 7. This experiment investigates the bandwidth allocation efficiency achieved by *MTRA*. Because *tree routing* is certain to find a *bandwidth-optimization VPN tree* for each request, we compare the average amount of bandwidth allocated for processing 100 requests in *MTRA* with *tree routing*. As the cost functions defined in Section 5, we expect the behavior of *MTRA* will be more similar to *tree routing* as the residual bandwidth amount on links ($B(l_x)$) is more abundant. We conducted 8 experiments for each amount of residual bandwidth on the links, and took the average on the allocated bandwidth in these 8 runs. For the comparison to be fair, only simulation runs that had no rejected requests were considered.

We define RS_{MTRA} and $RS_{Tree\,routing}$ as the average amount of bandwidth allocated for processing 100 requests in *MTRA* and *tree routing*, respectively. We also define $Percent_{Extra\_BW} = (RS_{MTRA}\text{-}RS_{Tree\,routing})/RS_{Tree\,routing}$. The simulation results are shown in Table 8. As expected, *MTRA* achieves better bandwidth efficiency when $B(l_x)$ is more abundant. For all the $B(l_x)$ values we consider in this experiment, the values of $Percent_{Extra\_BW}$ are all below 3%. Therefore, *MTRA* can achieve fairly good bandwidth allocation efficiency.

7.3. Running times

While *MTRA* achieves a lower *rejection ratio* in VPN provisioning, it has a longer running time than other approaches. In Table 9, we briefly list the average running times (in second) of the three provisioning algorithms for handling 100 random requests in the first four simulations. *MTRA* uses roughly 1 second for handling a VPN request for a network up to 120 links and 20 routers that the performance is acceptable. Note that all the simulations are executed on a notebook computer with 1.8 GHz Pentium-M CPU and 768 MB memory, and Microsoft XP OS.

8. Conclusions and future works

Several *hose-model* VPN provisioning algorithms have been proposed previously [3–5]. However, issues about the *rejection ratio* achieved by provisioning algorithms for establishing multiple VPNs on-line have not been investigated. In this paper, we show by concrete examples that all the provisioning algorithms proposed in [3–5] are unable to achieve a satisfactory *rejection ratio*

Table 9 The average running times on the three algorithms

Provisioning algorithm Simulation number	*MTRA*	*Tree routing*	*WSP provider-pipes*
Simulation 1	80.609	57.860	23.723
Simulation 2	133.799	87.312	25.169
Simulation 3	154.275	105.896	67.512
Simulation 4	122.738	80.882	17.453

in this case. To address the problem, we propose a new *hose-model* VPN provisioning algorithm called *MTRA*. We also derive the theoretical upper bounds of the *rejection ratios* achieved by *provider-pipes, tree routing* and *MTRA*, respectively. In addition, we set up an experimental environment, called *HVPAS*, to evaluate the performance of different *hose-model* VPN provisioning algorithms. According to the simulation results, *MTRA* can indeed effectively reduce the *rejection ratio*.

A number of issues related to *hose-model* VPNs still needs to be investigated. For example: (1) Designing a good label assignment schemes on the MPLS network for *MTRA* in order to minimize the number of labels needed; and (2) designing an efficient restoration algorithm under a single element (single node or single link) failure model. We will address these issues in our future work.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable comments that greatly helped improve the quality of this paper. This work was supported partially by National Science Council under grants NSC 92-2213-E-001-011 and NSC 93-2213-E-001-006.

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 and 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. 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.
6. A. Juttner, I. Szabo and Á Szentesi, On bandwidth efficiency of the hose resource management model in virtual private networks, in: *Proc. of IEEE INFOCOM* (2003).
7. D.O. Awduche, J. Malcom, J. Agobua, M. O'Dell and J. Mcmanus, Requirement for traffic engineering over MPLS, IETF RFC 2702, Sep. 1999.
8. 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.
9. A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi and B. Yener, Provisioning a virtual private network: A network design problem for multicommodity flow, in: *Proc. of the 33*rd*Annual ACM Symposium on Theory of Computing (STOC)* (2001).
10. C. Swamy and A. Kumar, Primal-dual algorithms for connected facility location problems, in: *Proc. of the 5*th *International Workshop on Approximation Algorithms for Combinatorial Optimization* (2002).
11. T.H. Cormen, C.E. Leiserson and R.L. Rivest, Introduction to algorithms, twentieth printing, MIT Press, 1998.
12. A. Balasubramanlan and G. Sasaki, Bandwidth requirement for protected VPNs in the hose model, in: *Proc. of IEEE International Symposium on Information Theory* (2003).

13. C.T. Chou, Traffic engineering for MPLS-based virtual private networks, Computer Networks 44(3) (2004) 319–333.
14. K. Kar, M. Kodialam and T.V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with mpls traffic engineering applications, IEEE J. Selected Areas in Communications 18(12) (2000) 2566–2579.
15. M. Kodialam and T.V. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, in: *Proc. of IEEE INFOCOM* (2000).
16. S. Suri, M. Waldvogel and P.R. Warkhede, Profile-based routing and traffic engineering, Computer Communications 26(4) (2003) 351–365.
17. B. Wang, X. Su and C.L. Philip Chen, A new bandwidth guaranteed routing algorithm for MPLS traffic engineering, in: *Proc. of IEEE International Conference on Communications* (2002).
18. Yi Yang, L. Zhang, J.K. Muppala and S.T. Chanson, Bandwidth-delay constrained routing algorithms, Computer networks 42(4) (2003) 503–520.
19. G. Apostolopoulos, R. Guérin, S. Kamat and S. K. Tripathi, Server-Based QoS Routing, in: *Proc. of IEEE GLOBECOM* (1999).
20. S. Plotkin, Competitive routing of virtual circuits in ATM networks, IEEE J. Selected Areas in Communications 13(6) (1995) 1128–1136.
21. E. Horowitz, S. Sahni and S. Anderson-Freed, Fundamentals of data structure in C, Computer Science Press, 1993.
22. A. Medina, A. Lakhina, I. Matta and J. Byers, BRITE: Universal topology generation from a user's perspective, http://www.cs.bu.edu/brite/publications/usermanual.pdf, April 2001.
23. A. Medina, A. Lakhina, I. Matta and J. Byers, BRITE: An approach to universal topology generation, in: *Proc. of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems(MASCOTS)*, (Aug. 2001).
24. R. Guerin, D. Williams and A. Orda, QoS routing mechanisms and OSPF extensions, in: *Proc. of IEEE GLOBECOM* (1997).