On the On-Line Hose-Model VPN Provisioning

Yu-Liang Liu[1], Yeali S. Sun[1] and Meng Chang Chen[2]
[1]*Dept. of Information Management National Taiwan University, Taiwan*
{d8725001,sunny}@im.ntu.edu.tw
[2]*Institute of Information Science, Academia Sinica, Taiwan*
mcc@iis.sinica.edu.tw

Abstract

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 flexible and convenient ways to specify the bandwidth requirement 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 *OHVPA* to address the issue. *OHVPA* 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 *OHVPA* performs better in *rejection ratio* than other provisioning algorithms.

1. Introduction

Traditionally, a private network (PN) is established by grouping dedicated lines connecting several geographically dispersed sites (endpoints). However, as the number of endpoints is growing, connecting them with dedicated lines is becoming increasingly expensive. As a result, Virtual Private Networks (VPNs) have emerged as replacements for PNs in recent years. VPN is a logical network that is established on top of a packet-switched network backbone with the goal of providing 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.

In the *hose model*, customers only need to specify the ingress bandwidth requirement, $b^-(v)$, and the 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 given time, and the value $b^+(v)$ is the maximum rate of traffic that endpoint v sends into the network at any given time. 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* [4,5], (2) *Hose-specific state* [4,5], (3) *VPN-specific state* [4,5], and (4) *Tree routing* [14]. For the rules for selecting a path between each endpoints pair and the allocated bandwidth in these algorithms, please refer to [4,5,12,14]. Our proposed algorithm can be implemented on a MPLS network, as the path pinning capacity provided by MPLS (multiprotocol label switching) technology can be used to direct the routing of an explicit path between each endpoint pair of a VPN [2,9].

In this paper, we consider the problem of minimizing the *rejection ratio* 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 paths of a VPN are determined, the service provider needs to explicitly allocate sufficient bandwidth on the links of the network backbone 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 next VPN provisioning. One of the requisites of a good on-line VPN provisioning algorithm is that it should achieve a low rejection ratio. However, previous hose-model VPN provisioning algorithms [4,5,14] have been unable to meet this requirement. We, therefore, propose a new provisioning algorithm, the *On-Line Hose-Model VPN Provisioning Algorithm* (*OHVPA*) to address this issue. Our experimental simulations show that the *OHVPA* can reduce the rejection ratio effectively. In addition, it can process multiple VPN setup requests rapidly as well. Given a network graph

*This work is partly supported by NSC under grants NSC 92-2213-E-002-088 and 93-2213-E-001-006. .

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 **IEEE**

G with n nodes and m edges, *OHVPA* spends only O(mn) time for a VPN setup request.

We summarize our contributions here. (1) Until now, issues about the rejection ratios achieved by on-line hose-model VPN provisioning algorithms have never been investigated. (2) We show by concrete examples that all of the *provider-pipes*, *hose-specific state*, *VPN-specific state* and *tree routing* algorithms are unable to achieve satisfactory rejection ratios. (3) We propose a new hose-model VPN provisioning algorithm called *OHVPA* to address this issue. (4) We derive the theoretical upper bounds of the rejection ratios for the *provider-pipes*, *tree routing* and *OHVPA* algorithm for the problem we consider in section 6 of [22].

2. Related Works

The *hose model* was first proposed by Duffield et al. in [4,5]. In their papers, *provider-pipes*, *hose-specific state* and *VPN-specific state* provisioning algorithms for hose-model VPNs were also proposed. Duffield et al's work inspired other researchers to develop provisioning algorithms for designing bandwidth-optimization hose-model VPNs. Kumar et al. argued that such VPNs should be based on a tree topology (hereafter called: *VPN tree*) [14]. They also presented an algorithm to compute the bandwidth-optimization *VPN tree* in the case 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 [8]. Swamy and Kumar further reduced the ratio to 5 [19]. In the case where the links on the network backbone have finite capacity, Gupta et al. proved that computing the bandwidth-optimization *VPN tree* is *NP-hard* [8].

Jűttner et al. compared the bandwidth efficiency of the hose model with that of the customer-pipe model [12]. They also conducted simulations to compare the bandwidth efficiency of the four hose-model VPN provisioning algorithms mentioned in section 1. Italiano et al. proposed an algorithm to make a hose-model VPN fault-tolerant under the case of singe link failure [11]. Balasubramanlan and Sasaki compared the bandwidth requirement of different hose-model VPN recovery algorithms by experimental simulations [3]. Gupta et al. investigated the issues about MPLS labels design and routing protocol for a *VPN tree* [9]. Erlebach and Rüegg proposed an optimal provisioning algorithm for hose-model VPN considering the case of multi-path routing between each endpoint pair of a VPN [6]. In multi-path routing, traffic between each endpoint (u,v) of a VPN can be split into multiple paths in an arbitrary manner. However, traffic in the application may be inherently un-splittable. We, therefore, consider the case of single-path routing in this paper.

3. Problem Formulation and Modeling

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

3.1 VPN Setup Request Modeling

The demands for VPN service by customers are described by VPN setup requests. In this paper, we consider that the bandwidth requirement of each endpoint e_j is symmetric. Let $b(e_j)$ denote the bandwidth requirement of endpoint e_j, and *Maxr* denote the maximum bandwidth guarantee provided by service providers. The ith VPN setup request, denoted by vr_i, describes a VPN that customers request service provider 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 access routers *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.

3.2. *On-line Hose-model VPNs Problem*

Although, the *OHVEP* defined in this paper is similar to the work in [15,18,20,21], which mainly consider on-line establishment of bandwidth guaranteed point-to-point tunnels. However, in the context of VPN provisioning, the basic unit concerned is a VPN consisting of numerous point-to-point tunnels, rather than a point-to-point tunnel, that makes the problem more challenging.

In *OHVEP*, service providers manage an MPLS network backbone G (described in subsection 3.1) on which VPNs are established. The VPN setup requests of customers are sent to VPN request server (described in the last paragraph of this subsection) by service provider. We consider the situation where (a) VPN setup requests arrive one by one independently and (b) information about future VPN setup requests

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 IEEE

is unknown. This information 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 service provider must process each VPN setup request in an on-line manner, as the off-line model in not suitable.

Upon receiving a VPN setup request, vr_i, the service provider triggers the provisioning algorithm to establish a corresponding VPN. The provisioning algorithm performs this task by first choosing a path between each endpoint pair and then allocating bandwidth on each link on the paths. 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 MPLS routing algorithms in [15,18,20,21] also use the rejection ratio (of tunnel setup requests) as the performance metric to compare different MPLS routing 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 this paper, we assume that service provider uses a server-based strategy [1] for processing VPN setup requests. In such a strategy, the VPN provisioning algorithm is run on a single entity called *VPN request server* (*VRS*), which keeps the complete link state topology database. It is also responsible for finding an explicit path for each endpoint pair of a VPN. Then the explicit 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 there exists a link state routing protocol for information acquisition.

4. Motivation for New Provisioning Algorithms

In this section, we elaborate on the reasons why the four provisioning algorithms proposed in [4,5,14] cannot achieve satisfactory *rejection ratios* under *OHVEP*.

Reason 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 relation holds for the bandwidth allocated on each link by different provisioning algorithms when establishing a VPN (the relation also holds for total bandwidth allocation):

$$BW_{\text{Provider-pipes}} \geq BW_{\text{Hose-specific}} \geq BW_{\text{VPN-specific}}$$ [12]

In addition, the simulations in [12] show that the allocated bandwidth of *tree routing* is less than that of *VPN-specific* to establish a VPN. In the case of establishing multiple VPNs, the difference between the allocated bandwidth 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 the following, we show that even the *tree routing algorithm* cannot guarantee a satisfactory *rejection ratio*.

Reason 2: (Ignorance of information about the amount of residual bandwidth for tree links selection in the *tree routing* algorithm will results in a higher rejection ratio):

We use an example to explain our argument. Suppose the service provider receive two VPN setup requests vr_1=(2,3,3) and vr_2=(3,3,3) as shown in figure 1. The residual bandwidth on all links in L is 5 units. 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 this example is 50%.

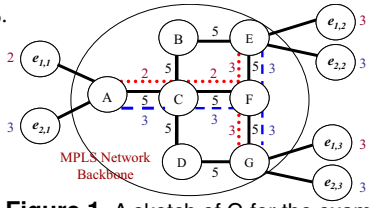

Figure 1. A sketch of G for the example

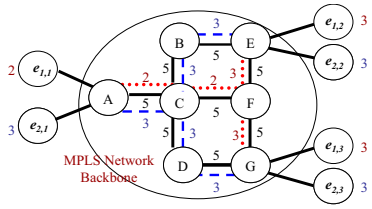

Figure 2. Optimal arrangement for the example

In fact, the amount of available resources on G is

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 IEEE

enough to accommodate both requests. If we rearrange the *VPN tree* of vr_2 as shown by the dashed lines in figure 2, then both of vr_1 and vr_2 can be accepted in this case. The rejection ratio achieved by this rearrangement is 0%. *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*, regardless of the amount of residual bandwidth on them. If the amount of residual bandwidth 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. Note that Kumar et al. also proposed a heuristic algorithm to compute a near-bandwidth-optimal *VPN tree* in the case of finite residual link capacity [13]. However, the residual bandwidth amount information is only for feasibility check of the VPN tree output by algorithm. Links that are thinly spread may also be chosen to form the VPN tree, and hence raise the likelihood of future requests rejection.

5. *OHVPA*

To address the problems described in section 4, we propose a new provisioning algorithm called the *On-Line Hose-Model VPN Provisioning Algorithm* (*OHVPA*). The design of *OHVPA* considers both bandwidth allocation efficiency and load balancing. Because of the excellent bandwidth allocation efficiency of tree topology for provisioning a single Hose-Model VPN shown by the experiments in [12], *OHVPA* also adopts VPN tree for establishing each VPN. The cost function of *OHVPA* for VPN tree selection is defined as following:

$$Cost_{OHVPA}(T) = \sum \frac{RS(l_x)}{B(l_x)},$$

where l_x is a link on a VPN tree T, and $RS(l_x)$ and $B(l_x)$ represent the amount of bandwidth needed and the amount of residual bandwidth, respectively. The cost function of *OHVPA* is inspired by the cost function defined in the routing algorithms proposed in [17,20] for route selection. The pseudo code of *OHVPA* is described below in figure 3.

***On-Line Hose-Model VPN Provisioning Algorithm* (*OHVPA*)**
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 $v\in N$
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 $l_x\in VT_{MC}$
13. {$B(l_x) = B(l_x)$-$RS(l_x)$;}
14. Accept(vr_i); Return(VT_{MC});
15. }

Figure 3. The pseudo code for *OHVPA*

When processing a request, *OHVPA* tries to find a *VPN tree* that minimizes the cost function defined above. It is clear that 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, *OHVPA* tries to finds a *VPN tree* that has abundant residual bandwidth and only requires a small amount of bandwidth to be allocated to the tree links. As a result, *OHVPA* can look after both bandwidth allocation efficiency and load balancing.

Given a network graph G consisting of n nodes. *OHVPA* iterates totally n times (once for each $v\in N$) to process a VPN setup request vr_i. In each iteration, *OHVPA* 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 do not exist any PT_v ($v\in N$) on which all links have enough residual bandwidth for allocation, *OHVPA* will reject vr_i. In the case of accepting vr_i, *OHVPA* 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, *OHVPA* 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, *OHVPA* first find a *BFS tree* (breadth first search tree [10]), T_v, rooted at v (by calling Function *BFS_Tree*). T_v contains all nodes in G and, in addition, it may contain nodes which are not VPN access routers used in vr_i as leaf nodes. Therefore, *OHVPA* prunes T_v and obtained a candidate *VPN tree* PT_v, on which all leave nodes are VPN access routers used in vr_i (by calling Function *Prune_Tree*).

OHVPA 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*). To compute the value of $RS(l_x)$ ($l_x\in T$), we first remove l_x from T, which

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 **IEEE**

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 $BR\_T_x^b$.

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

The time complexity of each iteration in *OHVPA* 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 *OHVPA* for processing a request is O(*mn*).

6. PERFORMANCE EVALUATIONS

For brevity, in this section, we only describe two simulations which compare the *rejection ratios* achieved by the *provider-pies*, *tree routing* and *OHVPA* algorithms for *OHVEP*. For more simulation results, please refer to section 7 in [22]. Note that simulation results for comparing average links utilization on *G* and bandwidth efficiency achieved by these algorithms are also presented there. In addition, theoretic upper bounds of rejection ratios achieved by these provisioning algorithms are derived in section 6 in [22].

Note that in both simulations, *K* denotes the total number of requests generated randomly, the number of endpoints of each VPN is generated randomly between 2 and *p*, and the bandwidth requirement of each endpoint is generated randomly between 1 to *Maxr*.

Simulation 1: (Performance Comparison in *KL topology*)

The parameter configuration of Simulation 1 is shown in Table 1. Due to extensive adaptation of the *KL topology* as MPLS network backbone in the literature about MPLS traffic engineering [15,18,20-22], we also adopt it as *G*. The *KL topology* is composed of 15 routers and 28 links. For a more detailed explanation of the *KL topology*, please refer to [15,18,20-22].

Table 1. Parameter configuration of Simulation 1

G	$B(l_i)$	*p*	*Maxr*	*K*
KL topology	Light links=1,500 units, dark links=6,000 units	7	75	100

We conduct 15 runs in this simulation, in each of which, 100 requests are randomly generated. The simulation results are shown in figure 4. The x-axis represents the run no., and the y-axis represents *rejection ratios* achieved by the provisioning algorithms in each run. We can see that the rejection ratio achieved by *OHVPA* is much lower than that achieved by the *provider-pipes* and *tree routing* algorithm. The rejection ratios achieved by *OHVPA* are 0% in all runs except in run 9 (where it is only 3%). However, the rejection ratios achieved by the *provider-pipes* and *tree routing* algorithms range from 35% to 55%. According to the simulation results, we believe that *OHVPA* can reduce the *rejection ratio* effectively in the *KL topology*.

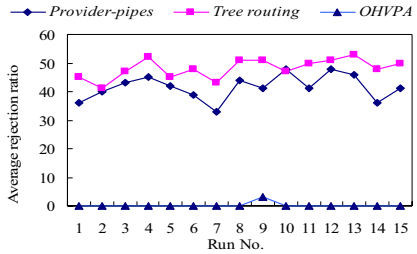

Figure 4. The rejection ratios in *KL Topology*

Simulation 2: (The rejection ratios on general *G*)

The parameter configuration of Simulation 2 is shown in Table 2. In order to evaluate the performance of *OHVPA* on general *G*, we use Brite[16] to randomly generate a connected 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 conduct 8 runs for each value of *Maxr*, and take the average *rejection ratio* achieved in these 8 runs.

Table 2. Parameter configuration of Simulation 2

G	$B(l_i)$	*p*	*Maxr*	*K*
Random generated by Brite with 20 nodes and 40 links	1,500 units	6	40~120 step 20	100

The simulation results are shown in figure 5. The x-axis represents the value of *Maxr*, and the y-axis represents average *rejection ratios* achieved by the provisioning algorithms. As expected, in all the three algorithms, the average *rejection ratio* increases as the value of *Maxr* increases. Even so, the average *rejection ratio* achieved by *OHVPA* is much lower than that achieved by the other two algorithms for all *Maxr* values, except for the light load case (*Maxr*=40), where the average rejection ratios is 0% in all the three algorithms. Even in the most heavily loaded case (where *Maxr*=120), the average rejection ratio achieved by *OHVPA* is only 10.125%; however, for the *provider-pipe* and *tree routing* algorithm, it is 44.5% and 36.75%, respectively. The experimental

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 IEEE

results show that *OHVPA* can indeed achieve a lower *rejection ratio* on general *G* compare to the other two algorithms.

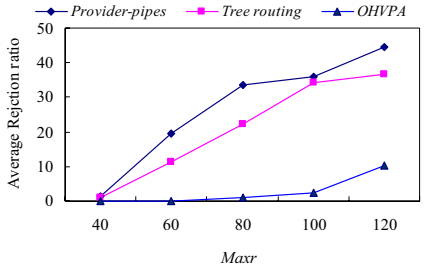

Figure 5. The Effect of Maxr

7. CONCLUSIONS

Several hose-model VPN provisioning algorithms have been proposed previously [4,5,14]. However, issues about the rejection ratio achieved by provisioning algorithms for establishing multiple VPNs on-line have never been investigated.

In this paper, we show by concrete examples that all the algorithms proposed in [4,5,14] are unable to achieve a satisfactory rejection ratio in this case. To address the problem, we propose a new hose-model VPN provisioning algorithm called *OHVPA*. We also derive the theoretical upper bounds of the rejection ratios achieved by the *provider-pipes*, *tee routing* and *OHVPA* algorithm, respectively. In addition, we have conducted experimental simulations to evaluate the performance of different hose-model VPN provisioning algorithms. According the simulation results, *OHVPA* can indeed reduce the rejection ratio effectively.

References

[1] G. Apostolopoulos, R. Guérin, S. Kamat, S. K. Tripathi, Server-based QoS routing, *IEEE GLOBECOM*, 1999.

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

[3] A. Balasubramanlan and G. Sasaki, Bandwidth Requirement for Protected VPNs in the Hose Model, in: *Proc. of IEEE International Symposium on Information Theory* 2003.

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

[5] 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, vol. 10, no. 5, pp. 565-578, October 2002.

[6] T. Erlebach and M. Rüegg, Optimal Bandwidth Reservation in Hose-Model VPNs with Multi-Path Routing, in Proc. of IEEE INFOCOM, 2004.

[7] R. Guerin, D. Williams and A. Orda, QoS Routing Mechanisms and OSPF extensions, in Proc. of IEEE GLOBECOM, 1997.

[8] 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 33rd Annual ACM Symposium on Theory of Computing (STOC), 2001.

[9] 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.

[10] E. Horowitz, S. Sahni and S. Anderson-Freed, Fundamentals of Data Structure in C, Computer Science Press, 1993.

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

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

[13] A. Kumar, R. Rastogi, A. Silberschatz and B. Yener, Algorithms for Provisioning Virtual Private Networks in the Hose Model, Bell Labs, Tech. Memo., 2000.

[14] A. Kumar, R. Rastogi, A. Silberschatz and B. Yener, Algorithms for Provisioning Virtual Private Networks in the Hose Model, IEEE/ACM Transactions on Networking, vol. 10, no. 4, August 2002.

[15] 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, vol. 18, no. 12, pp.2566-2579, December 2000.

[16] 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.

[17] S. Plotkin, Competitive Routing of Virtual Circuits in ATM Networks, IEEE J. Selected Areas in Communications, vol. 13, no. 6, pp.1128-1136, August 1995.

[18] S. Suri, M. Waldvogel and P. R. Warkhede, Profile-Based Routing and Traffic Engineering, Computer Communications, 24(4), pp. 351-365, March 2003.

[19] C. Swamy and A. Kumar, Primal-Dual Algorithms for Connected Facility Location Problems, Proc. of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization, 2002.

[20] 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.

[21] Yi Yang, L. Zhang, J. K. Muppala and S. T. Chanson, Bandwidth-delay Constrained Routing Algorithms, Computer Networks, vol. 42, issue 4, pp. 503-520, July 2003.

[22] Yu-Liang Liu, Yeali S. Sun and Meng Chang Chen, "*OHVPA*: An On-Line Hose-Model VPN Provisioning Algorithm", Technical report TR-IIS-04-020, IIS, Academia Sinica, 2004, available at: http://www.iis.sinica.edu.tw/LIB/TechReport/tr2004/tr04020.pdf.

Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop
0-7695-2388-9/05 $20.00 © 2005 **IEEE**