Hierarchical Provisioning Algorithm for Virtual Private Networks Using the Hose Model

Monia Ghobadi, Sudhakar Ganti, Gholamali C. Shoja
University of Victoria, Victoria BC, Canada V8W 3P6
e-mail: *{monia, sganti, gshoja}@cs.uvic.ca*

***Abstract*— Virtual Private Networks (VPN) provide a secure and reliable communication between customer sites over a shared network. Two models were proposed for the service provisioning in VPNs. The "hose model" for VPNs alleviates the scalability problem of the "pipe model" by reserving bandwidths for aggregate ingress and egress requirements instead of between every pair of VPN endpoints. In this work, VPN endpoints are connected using a tree structure and our algorithm optimizes the total bandwidth reserved on edges of the VPN tree. We introduce a fast and efficient algorithm in finding the shared VPN tree to reduce the total provisioning cost. Our simulation results indicate that the VPN trees constructed by our proposed algorithm reduce bandwidth requirements as compared to previously proposed algorithms while having a much smaller execution time.**

Keywords- Virtual Private Networks; Hose Model; Quality of Service; Provisioning Cost; Spanning Tree.

I. INTRODUCTION

A Virtual Private Network (VPN) is a group of computer systems connected as a private network that communicates over a public network. The aim is to provide the VPN endpoints with a service comparable to a dedicated private network established with *leased lines*. Thus, providers of VPN services need to address the QoS and security issues while deploying a VPN over a shared IP network. In recent years, substantial progress in the IP security technologies have enabled existing VPN service offerings to provide customers with a level of privacy comparable to that offered by a dedicated line [5]. The emergence of IP technologies such as MPLS and RSVP-TE [1] have made it possible to realize IP-based VPNs that can provide the end customers with QoS guarantees. In this paper, we address the problem of resource allocation in VPN hose model with QoS guarantees while optimizing total provisioning cost.

Two popular models have been proposed for providing QoS in the context of VPNs: the "pipe" model [1] and the "hose" model [2]. As depicted in Fig. 1, in the pipe model, a VPN customer buys a set of customer-pipes. In the "hose" model, as illustrated in Fig. 2, each VPN endpoint connects to the network by a hose, which is specified by its aggregate ingress and egress bandwidth requirements. The hose model has desirable characteristics such as ease of specification, flexibility, and multiplexing gain [2]. A number of provisioning algorithms for VPNs in the hose model have been proposed [2, 3, 5, 6, 10, 15]. Further, in [5], it has been shown that optimal bandwidth allocation problem in VPN

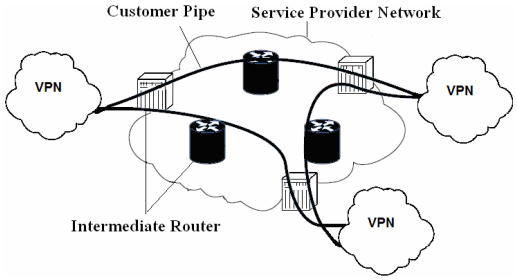

Figure 1. VPN pipe model

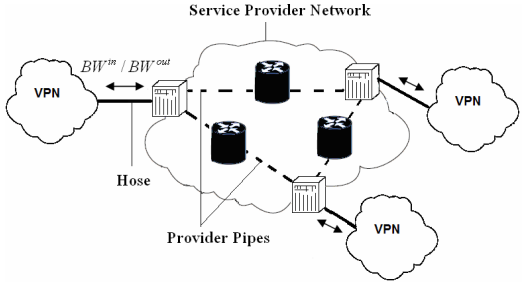

Figure 2. VPN hose model

hose model is NP-hard. The bandwidth efficiency of the hose model is studied in [3] where the over-provisioning factor of the model is evaluated in networks with various sizes and node densities. In [15], we proposed a new ranking approach to enhance the hose model to guarantee delay requirements between endpoints while optimizing the provisioning cost. In this paper, we present a new hierarchical approach, called HIST algorithm, for optimal resource provisioning in the VPN hose model. Our HIST algorithm is more efficient in terms of time complexity and provisioning cost than the one used in [15].

The rest of this paper is organized as follows. A short review of the previous works is presented in Section II. The Hierarchical Iterative Spanning Tree (HIST) algorithm that minimizes the provisioning cost is described in Section III. Section IV presents the discussion on time complexity of the algorithm. Simulation results studying the performance of the proposed algorithm are presented in Section V. Finally, Section VI concludes the paper.

II. MOTIVATION AND RELATED WORK

Although the hose model provides customers with simpler and flexible Service Level Agreements (SLA), the model presents the provider with a challenging problem of resource

management. The main problem of interest in this work is that of constructing a shared tree connecting all the VPN endpoints with the objective of minimizing the provisioning cost. Compared to the pipe and source-based tree approaches, the shared tree approach makes the best use of statistical multiplexing to reduce the provisioning cost. Thus, we consider tree structures to connect the VPN endpoints since trees are scalable and simplify routing and restoration. Furthermore, trees allow the bandwidth reserved on a link to be shared by the traffic between the two sets of VPN endpoints connected by the link [5].

The primal-dual algorithm for computing VPN tree was developed by Kumar et al. [5]. Their approach finds the near optimal provisioning tree in which a 10- approximation is obtained by solving a linear program relaxation, and rounding the fractional solution. This approach can find a tree with cost less than a Steiner tree or BFS tree [5]. In that work, a VPN network is modeled as a graph $G = (V, E)$ where V is the set of nodes and E is the set of bidirectional links connecting the nodes. The VPN specification in the hose model includes: A subset of end points $P \subseteq V$ corresponding to the VPN endpoints; and for each VPN endpoint $i \in P$, the associated ingress and egress bandwidths B_i^{in} and B_i^{out}, respectively. The problem of computing the optimal VPN tree is formulated as the following:

Optimal Bandwidth-constrained Shared Tree Problem (*OBSTP*)*:* Given a set of VPN endpoints P and their ingress and egress bandwidths, find a shared tree T connecting VPN endpoints for which the total bandwidth reserved on edges of T is minimum.

It is proved in [5] that OBSTP is NP-hard and the authors suggested a primal-dual method to solve the problem. In the following section we explain our Hierarchical Iterative Spanning Tree (HIST) algorithm as a heuristic approach to find a near-optimal solution for the OBSTP. Our simulation results with synthetic network graphs as well as real Tier-1 ISPs indicate that the VPN trees constructed by our proposed algorithm require less bandwidth compared to primal-dual algorithm. Furthermore, we implemented and executed both algorithms on the same platform and the HIST algorithm's execution time is measured to be far less than that of primal-dual algorithm. The simulation results are discussed in more detail in Section V.

III. HIERARCHICAL ITERATIVE SPANNING TREE ALGORITHM

In our approach, we considered a two level network hierarchy: the core of the network and the edge of the network. VPN endpoints are located in the edge network and are connected to the routers in the core network. The edge network representing VPN endpoints for a particular customer is essentially different branches of that VPN. Our algorithm consists of two steps: step one is executed on the edge network to find a possible minimum cost tree connecting all the VPN endpoints without considering any intermediate routers in between. The result of this step is independent of the underlying network topology and is only dependent on the VPN endpoints' ingress and egress bandwidths. In step two we extend the result of step one to the core network and connect the VPN endpoints by intermediate routers in a way to reduce the provisioning cost.The idea of step one is to assume that all VPN endpoints are connected to each other as vertices of a graph G'. The graph G', which is constructed iteratively in this step, can be considered as a "virtual topology" in which VPN endpoints are connected by "virtual links". Thus, in this step, we try to find minimum cost shared tree $T_{G'}$ connecting the vertices in graph G' (the VPN endpoints). Later, in the second step, we replace each virtual edge (u, v) in $T_{G'}$ by the appropriate physical path between VPN endpoints u and v trying to keep the provisioning cost minimum.

Fig. 3 contains the ITERATIVE_SPANNING_TREE procedure which builds graph G' and outputs $T_{G'}$. The input of this procedure is the set of VPN endpoints P and the output is $T_{G'}$ that is the tree connecting VPN endpoints by virtual links. Since only the ingress and egress bandwidths of the VPN endpoints contribute to the shared tree's cost, this procedure only iterates on the VPN endpoints while the primal-dual algorithm iterates over all the nodes of the graph. As the number of VPN endpoints is normally 10 percent of the total number of nodes, this will reduce the execution time of our algorithm compared to primal-dual algorithm. Without loss of generality, assume that the VPN endpoints are indexed as p_1, p_2,…, $p_{|P|}$. The procedure starts with empty G' and $T_{G'}$ topologies. At iteration k, there is a tree $T_{G'}$ with k vertices connecting k VPN endpoints. At iteration k+1, the $(k+1)^{th}$ VPN node will eventually join the tree by adding node p_{k+1} to G' and also k edges from p_{k+1} to nodes p_1, p_2…, p_k in G'. To find the spanning tree $T_{G'}$ in G', we used a modification of the algorithm proposed by Shioura et al. in [12], recognized as the best algorithm in terms of the time complexity and memory requirements to compute all the spanning trees of a given graph.

Fig. 4 contains the MODIFIED_SHIOURA and FIND_CHILDREN procedures based on Shioura et al.'s all-spanning-trees and find-children procedures provided in [12]. The input of MODIFIED_SHIOURA procedure is graph G' and the output is *minTree* which is a spanning tree in G' with minimum provisioning cost over the enumerated spanning trees. Similar to find-children procedure in [12], calling FIND_CHILDREN procedure with arguments T^p, k, and *minTree* results in finding children of tree T^p not containing an edge e_k and saving the child with minimum cost in *minTree*. Our modifications to Shioura et al.'s algorithm includes adding lines 4-6 to FIND_CHILDREN procedure in order to find *minTree* as the tree with minimum cost over the enumerated spanning trees in graph G'. Lines 4 and 5 keep track of the tree with minimum cost and line 6 is a pruning scheme in which we find the children of a tree provided that the tree diameter, which is the longest shortest path between tree endpoints based on the number of hops, is less than its parent's diameter. As shown in [7], our pruning scheme helps in decreasing the number of enumerated spanning trees and hence the execution times, while it keeps the results close to the case without using the pruning scheme.

Fig. 5 contains the HIERARCHICAL_EXTENSION procedure. The input to this procedure is G and tree $T_{G'}$ which is the output of the last iteration of ITERATIVE_SPANNING_TREE procedure. The main goal of this procedure is to extend $T_{G'}$ to the core of the network to contain the intermediate routers. At the beginning of the procedure, the final shared tree

ITERATIVE_SPANNING_TREE(P)

1. $T_{G'} = \emptyset$, $G' = \emptyset$
2. **for** each vertex $v \in P$
3. $\quad G' \leftarrow T_{G'}$
4. add new vertex v to G'
5. add an edge from v to all other nodes in G'
6. $T_G \leftarrow$ MODIFIED_SHIOURA (G')
7. **return** ($T_{G'}$)

Figure 3. Algorithm for computing the virtual topology spanning trees

MODIFIED_SHIOURA (G')

1. $n \leftarrow$ number of nodes in G'
2. **if** $n \leq 2$ **return** G'
3. $minTree \leftarrow \emptyset$
4. $T^0 \leftarrow$ A depth-first spanning tree in G'
5. $minTree \leftarrow T^0$
6. $minTree \leftarrow$FIND_CHILDREN(T^0, n-1, $minTree$)
7. **return** ($minTree$)

FIND_CHILDREN (T_p, k, $minTree$)

1. **if** $k \leq 0$ **return** $minTree$
2. **for** each edge $g \in Entr(T_p, e_k)$ as defined in [12]
3. new tree T_c = Replace g with e_k in T_p
4. $\quad$ **if** cost of T_c < cost of $minTree$
5. $\quad\quad minTree \leftarrow T_c$
6. $\quad$ **if** diameter of T_c < diameter of T_p
7. $\quad\quad$ FIND_CHILDREN(T_c, k-1, $minTree$)
8. FIND_CHILDREN(T_p, k-1, $minTree$)

Figure 4. Algorithms for computing the locally minimum diameter spanning tree

HIERARCHICAL_EXTENSION ($T_{G'}$, G)

1. $finalTree = \emptyset$
2. **for** each edge $e \in G$
3. $\quad$ weight(e) $\leftarrow 1$
4. **for** each edge $(u, v) \in T_{G'}$
5. $\quad$ **if** (there is no path between u and v in $finalTree$)
6. $\quad\quad path_{uv} \leftarrow$ shortest path between u and v in G based on edge weights
7. $\quad\quad$ **for** each edge $g \in path_{uv}$ and $g \notin finalTree$
8. $\quad\quad\quad$ add g to $finalTree$
9. $\quad\quad\quad$ weight(g) $\leftarrow 0$
10. **return**($finalTree$)

Figure 5. Algorithm for replacing the virtual links by appropriate physical paths

connecting all the VPN endpoints in the network, *finalTree*, is empty and all edges in the network have weights equal to one. These weights will be used by the Dijkstra's algorithm [13]. For each edge (u, v) in $T_{G'}$, if there is no path between u and v in the *finalTree* already, we use Dijkstra's algorithm to find the shortest path between u and v in the graph G. The new edges will be added to the *finalTree*. Moreover, to increase the link sharing probability, we set the weights of all edges in G that were added to *finalTree* to zero. Thus, the edges that are already in *finalTree* have less weight and hence higher probability of being selected in Dijkstra's algorithm over other edges. This is done to increase the probability of using the current edges in *finalTree* which increases the probability of having fewer edges in the *finalTree* and reducing the total provisioning cost. Finally, when all the VPN endpoints are connected to each other, the resulting *finalTree* is the shared tree connecting all the VPN endpoints.

As an example, consider the network graph in Fig. 6(a). The four VPN endpoints 1, 2, 3, and 4 have ingress/egress bandwidth requirements of 3/12, 12/15, 5/8, 9/4 units respectively. Figures 6(b) to 6(f) depict the steps of performing the ITERATIVE_SPANNING_TREE procedure. Fig. 6(b) is the result of first iteration of the procedure, as there are only two nodes in the virtual topology. Figures 6(c) and 6(e) show the result of adding node 3 and 4 to G' respectively. Figures 6(d) and 6(f) depict the result of MODIFIED_SHIOURA procedure. The final result of ITERATIVE_SPANNING_TREE procedure is the tree $T_{G'}$ depicted in Fig. 6(f) which shows the optimum virtual topology to connect the VPN endpoints.

The results of performing the HIERARCHICAL_EXTENSION procedure are depicted in Figures 6(g) to 6(k). Fig. 6(g) shows the original network graph with each link having weight equal to 1. In Fig. 6(h) the bold links show the shortest path between nodes (1,3). These edges will be added to the *finalTree* and their weights will be set to 0. Figures 6(i) and 6(j) show the result of finding the shortest path between nodes (1,2) and nodes (1,4) respectively. The *finalTree* is shown in Fig. 6(k) with total provisioning cost of 68 bandwidth units. Although the total number of spanning trees of the network is 48, our hierarchical approach builds only 9 spanning trees to find the tree with minimum provisioning cost.

IV. COMPLEXITY ANALYSIS

In this section we will show that the time complexity of HIST algorithm is $O(p^3 + +pm+np\log n)$ where m is the number of edges, n is the number of nodes and p is the number of VPN endpoints in the network.

The loop in line 2 of ITERATIVE_SPANNING_TREE procedure performs p iterations, one for each VPN node. Further, for each iteration k of the loop, the MODIFIED_SHIOURA procedure is executed on a graph with k vertices and $2k$-3 edges. According to [12], the time complexity of MODIFIED_SHIOURA algorithm is $O(N_k+k+2k-3)$, where N_k is the number of spanning trees that the algorithm iterates for a graph with k vertices. Thus, the total time complexity of step 1 is $O(\sum_{k=1}^{p}(N_k+3k-3))=O(\sum_{k=1}^{p}N_k+p^2)$. We showed in [7] that using our pruning scheme will keep N_k to be $O(k^2)$ and hence the time complexity of step 1 is $O(\sum_{k=1}^{p}k^2+p^2)=O(p^3)$. The time complexity of HIERARCHICAL_EXTENSION procedure can be shown to be $O(p(m + n\log n))$.The reason being that the tree $T_{G'}$ would have p-1 edges and in the worst case the Dijkstra's algorithm time complexity is $O(m+n\log n)$ [13]. Thus our HIST algorithm has an over all time complexity of $O(p^3+pm+np\log n)$ which is far less than the time complexity of primal-dual algorithm that was given as $O(n(m^2p+mnp+n^2\log n))$ [5].

V. SIMULATION STUDY

In our simulations, we used two sets of network topologies. The first set is taken from the Rocketfuel project

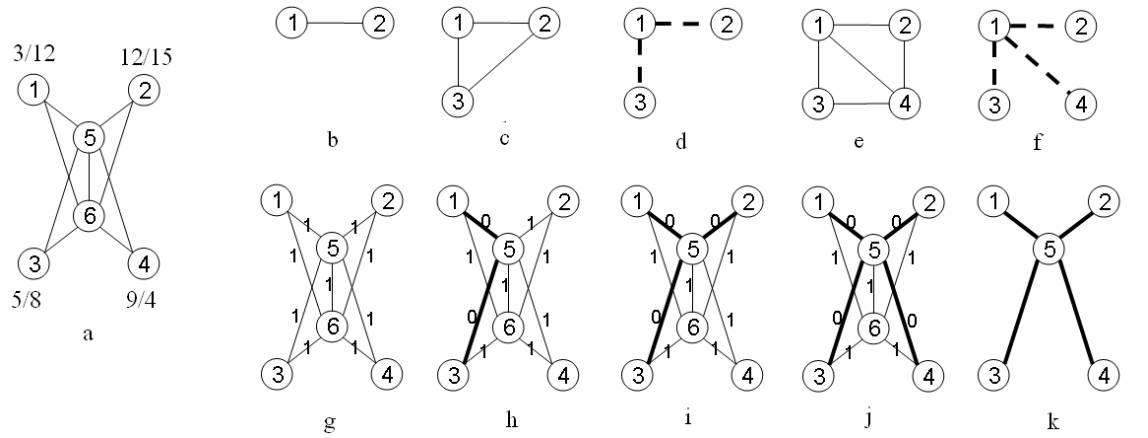

Figure 6. The results of performing HIST algorithm on a six node network with four VPN endpoints

[9]. Among all the topologies, we selected two dominant Tier-1 ISP topologies as listed in Table I.

The second set was randomly generated using the Waxman Model [4]. Since we can easily control the size of the topologies, we use them to study the effect of the network size. In this model, the nodes are placed on a 3000×2400KM^2 plane, roughly the size of the USA. The probability for two nodes to be connected by a link decreases exponentially with the Euclidean distance between them according to the following probability function: $P_e^{(u,v)} = \alpha \exp(-l(u,v)/L\beta)$ where L is the maximum distance between any two nodes in the network and $l(u, v)$ is the distance between u and v. The parameter β controls the ratio of short links to long links, while the parameter α controls the average node degree of the network. In our simulations, α and β were set at 2.2 and 0.15 respectively. These values were selected carefully to obtain random networks with close resemblance to real networks.

For both sets of topologies, the VPN endpoints were randomly selected from the network nodes. The number of VPN endpoints was set to 10% of the total number of network nodes in the network unless explicitly specified. The bandwidth requirement of each VPN endpoint was uniformly chosen between 2 and 100 Mbps. An asymmetry parameter is associated with each endpoint, representing the ratio between the ingress and egress bandwidth requirements. This ratio varies from 1 to 256 in our simulations. Each simulation result given below is the average of five rounds of simulation run. We calculated the 95% confidence intervals as $\hat{\theta} - t_{\alpha/2,f}\hat{\sigma}(\hat{\theta}) \le \theta \le \hat{\theta} + t_{\alpha/2,f}\hat{\sigma}(\hat{\theta})$ where $\hat{\theta}$ is the average value of simulations runs, $\hat{\sigma}^2(\hat{\theta})$ is the standard deviation and $t_{0.025,4}$ is 2.78 according to Table A.5 in [14]. The results show that the confidence intervals for provisioning costs are less than 0.5 Gbps.

Fig. 7 compares the provisioning cost of HIST algorithm and primal-dual algorithm [5] with the optimal solution. The optimal solution is found by constructing all the spanning trees of each network and finding the tree with minimum cost. The number of VPN endpoints in each network is fixed at 50 % of the total number of nodes. It can be observed that the total provisioning cost of HIST algorithm is very close to that of the optimal solution. Moreover, Fig. 8 compares provisioning cost of primal-dual algorithm with HIST algorithm for large random networks and real ISP topologies. The results show that our HIST algorithm requires less provisioning cost over all considered topologies. Fig. 9 shows that the execution time of HIST algorithm is far less than that

TABLE I. ROCKETFUEL ISP TOPOLOGIES USED IN THE SIMULATIONS

AS Number	Name	Tier	No.of Edges	No.of Nodes
1239	Sprint(US)	1	168	52
7018	ATT(US)	1	296	115

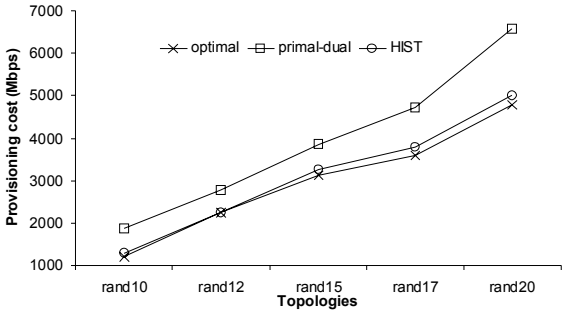

Figure 7. Provisioning cost comparison between HIST, primal-dual and Optimal solution

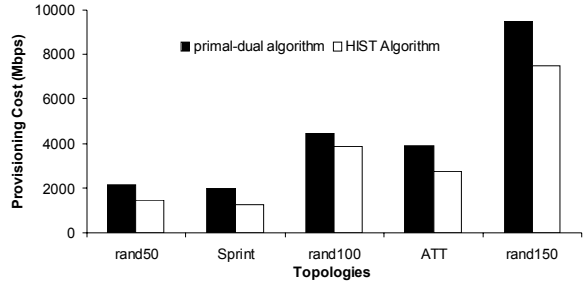

Figure 8. Provisioning cost comparison between primal-dual algorithm and HIST algorithm

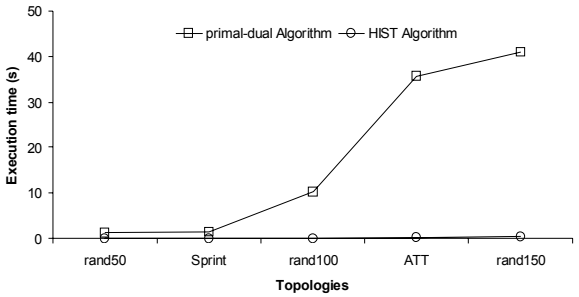

Figure 9. Execution time comparison between primal-dual algorithm and HIST algorithm

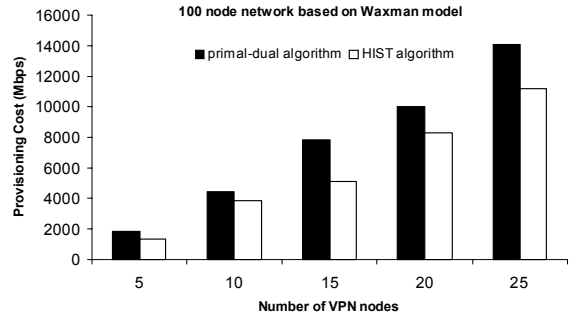

Figure 10. Effect of number of VPN nodes on provisioning cost

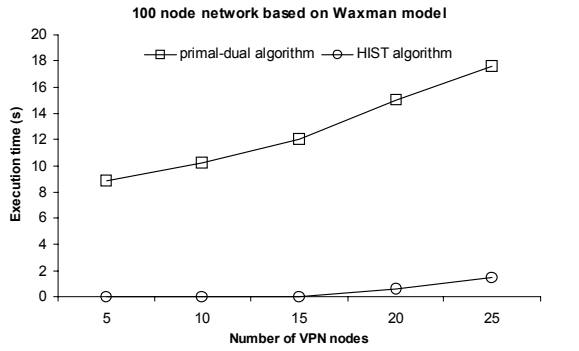

Figure 11. Effect of number of VPN nodes on execution time

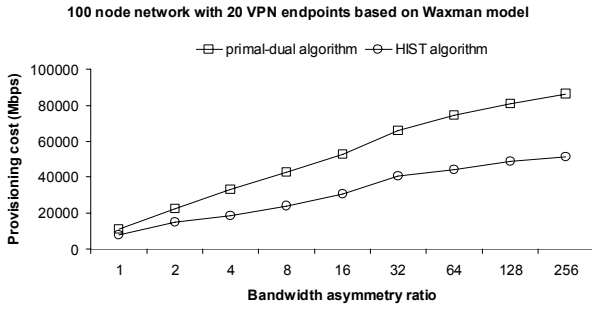

Figure 12. Effect of asymmetry ratio on provisioning cost

of the primal-dual algorithm since the former iterates over VPN endpoints while the latter iterates over all network nodes. Figures 10 and 11 illustrate the effect of increasing number of VPN endpoints on total provisioning cost and execution time for a 100 node network based on Waxman model, respectively. The results show that the HIST algorithm finds a tree with smaller with low execution time than primal-dual algorithm.

Fig. 12 studies the effect of changing the bandwidth asymmetry ratio on provisioning cost for a 100 nodes network with 20 VPN endpoints. The ratio between ingress and egress bandwidth of VPN endpoints has been increased from 1 to 256. The results show that our HIST algorithm would still perform better than the primal-dual algorithm.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we introduced a new Hierarchical Iterative Spanning Tree (HIST) algorithm to optimize the provisioning cost in the VPN hose model. This scheme will result in lower provisioning costs than the previous work introduced in [5]. Our simulation results with synthetic network graphs as well as real Tier-1 ISPs indicate that the VPN trees constructed by HIST require lower bandwidth reservation when compared to primal-dual algorithm [5]. Furthermore, our HIST algorithm's execution time is measured to be far less than that of the primal-dual algorithm.

REFERENCES

[1] B. Davie, Y. Rekhter. "MPLS Technology and Applications", San Mateo, CA: Morgan Kaufmann, 2000.

[2] N. G. Duffield, P. Goyal,A. Greenberg, P. Mishra, K. K Ramakrishnan, J. E. van der Merwe, "A flexible model for resource management in Virtual Private Networks", In Proc. ACM SIGCOMM, vol 29(4), 1999, pp. 95-108.

[3] A. Juttner, I. Szabo, A. Szentesi, "On bandwidth efficiency of the Hose resource management model in Virtual Private Networks", In Proc. INFOCOM, vol. 1, 2003, pp. 386-395.

[4] B. M. Waxman, "Routing of multipoint connections", IEEE Journal on Selected Areas in Communications, vol. 6(9), 1988, pp. 1617-1622.

[5] A. Kumar, R. Rastogi, A. Silberschatz, B. Yener, "Algorithms for provisioning Virtual Private Networks in the hose model", IEEE/ACM Transaction on Networking, vol. 10(4), 2002, pp. 565-578.

[6] G. de Veciana, S. Park, A. Sang, S. Weber. "Routing and provisioning VPNs based on hose traffic models and/or constraints". In Proc. 40th Annual Allerton Conference on Communication Control and Computing, 2002, pp. 77-86.

[7] M. Ghobadi, M.Sc. Thesis, "Resource Optimization Algorithms for Virtual Private Networks Using the Hose Model", Department of Computer Science, University of Victoria, BC, Canada, 2007.

[8] L. Zhang; J. Muppala, S. Chanson, "Provisioning virtual private networks in the hose model with delay requirements." Hong Kong University, International Conference on Parallel Processing, 2005, pp.211 – 218.

[9] Rocketfuel project, Computer Science and Engineering, Univ. of Washington. http://www.cs.washington.edu/research/networking/rocketfuel

[10] A. Gupta, A. Kumar, Kleinberg, R. Rastogi, B.Yener, "Provisioning a Virtual Private Network: A network design problem for multicommodity flow", In Proc. ACM STOC, 2001, pp. 389-398.

[11] S. Hakimi. "Optimal locations of switching centers and medians of A graph", Operations Research, vol. 12, 1964, pp. 450-459.

[12] A. Shioura, A. Tamura, T. Uno, An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graph, SIAM J .Comput. 26(3): 678-692 (1997).

[13] Dijkstra, E. W. "A Note on Two Problems in Connection with Graphs." Numerische Math. 1, 269-271, 1959.

[14] J. Banks, J. S. Carson, B. L. Nelson, D. M. Nicol, "Discrete-Event System Simulation", Fourth Edition, Prentice Hall, 2005.

[15] M. Ghobadi, S. Ganti, G.C. Shoja, "Resource Optimization to Provision a Virtual Private Network Using the Hose Model", In Proc. IEEE International Conference on Communications, 2007.