International Journal of Management Science
Vol 12, No 2, November 2006

A Layered Network Flow Algorithm for the Tunnel Design Problem in Virtual Private Networks with QoS Guarantee

Sang Hwa Song*

Graduate School of Logistics, University of Incheon
994-52, Songdo Dong, Yon Su Gu, Incheon, 406-130, Korea

Chang Sup Sung

Department of Industrial Engineering, KAIST
Gusong Dong, Daejon, 305-701, Korea

(Received Mar. 2006; Revised Sep. 2006; Accepted Sep. 2006)

ABSTRACT

This paper considers the problem of designing logical tunnels in virtual private networks considering QoS guarantee which restricts the number of tunnel hops for each traffic routing. The previous researches focused on the design of logical tunnel itself and Steiner-tree based solution algorithms were proposed. However, we show that for some objective settings it is not sufficient and is necessary to consider both physical and logical connectivity at the same time. Thereupon, the concept of the layered network is applied to the logical tunnel design problem in virtual private networks. The layered network approach considers the design of logical tunnel as well as its physical routing and we propose a modified branch-and-price algorithm which is known to solve layered network design problems effectively. To show the performance of the proposed algorithm, computational experiments have been done and the results show that the proposed algorithm solves the given problem efficiently and effectively.

Keywords: Layered Network Flow, Branch-and-Price Algorithm, Virtual Private Network, Network Optimization

1. INTRODUCTION

Virtual Private Network (VPN) service uses a nonprivate data network for its private traffic [9]. Since the VPN service appears to offer an alternative to the traditional leased line network by utilizing the established public network, it

* Corresponding author, Email : songsh@incheon.ac.kr

should provide a comparable service at very least cost. Therefore, it has been noted that VPN service providers should consider the issues of data security and network connectivity [7]. Substantial progress in technology for Internet security has provided us with the same level of the existing security and privacy provided based on the leased lines. However, much less attention has been paid to the issues of establishing network connectivity between nodes in VPN where the main mechanism for establishing such connectivity is creation of VPN tunnels. Since the traffic carried within the VPN is routed through any established VPN tunnels, the issue of how to establish such tunnels at minimum cost, while guaranteeing quality-of-service (QoS) to customers, is critical for successful VPN service provisioning, which provides the motivation of this paper to consider optimal tunnel design problem for the VPN service.

The layout of VPN tunnels may be optimized to satisfy many different optimization objectives. However, it has been noted that, in networks of today and in most of future networks, the cost of operation and maintenance of virtual topologies (VPN tunnels) may be larger than the cost of bandwidth [7]. Therefore, this paper focuses on minimizing the cost of operation and maintenance of virtual topologies, which strongly depends on both the network load and VPN tunnel maintenance cost. *Load* on a physical arc is defined as the cardinality of the VPN tunnels that share the physical arc (physical link), which equals to the number of entries in the routing table at the associated switching node. The network load represents the maximum value of the loads on the physical arcs in the network. It is noted that the network load should be minimized to simplify tunnel set up and management, provide the switching service for data at very high speed and design a good fault-tolerant layout [11, 12, 20]. If the network load increases, the signaling system for maintaining the virtual topology may break down [1]. Also the maintenance cost of each VPN tunnel is another important optimization parameter which indicates the cost of building and maintaining a VPN tunnel over each physical arc [7, 20]. This overhead is incurred for each VPN tunnel, and a physical arc that participates in several VPN tunnels encounters this overhead for each VPN tunnel independently of the other VPN tunnels. Therefore, the cost can be determined based on the weight associated with each physical arc. It is noted that the weight associated with each physical arc is determined based on administrative considerations of the core network operator and on the bandwidth allocated over this arc. Generally, over-used physical arcs are expected to be more costly than under-used physical arcs. Therefore, given the physical network topology and terminal nodes (VPN endpoints), the objective of the proposed **VPN Tunnel design (*VPNTD*) problem** is to establish the set of VPN tunnels (logi-

cal arcs) required to route the traffic demands between terminal nodes and determine the sub-route (physical path) of each VPN tunnel subject to VPN tunnel hop restriction.

The routing structure in VPN is hierarchical that the route of each traffic demand may be viewed as a concatenation of multiple VPN tunnels (sub-routes), while each VPN tunnel (a temporarily-established logical link between nodes) may be viewed as a concatenation of multiple physical links and used for simplifying network resource management as in Figure 1. To establish this network connectivity through VPN tunnels between VPN endpoints, various solution approaches were proposed [7, 10, 18-21]. Cohen and Kaempfer [7] have proposed Steiner-tree-based solution heuristics for the VPN tunnel design problem, which minimizes the operation and maintenance cost of VPN tunnels. Duffield *et al.* [10] have also suggested that a Steiner tree can be employed to connect VPN endpoints. However, even though a Steiner tree has the smallest number of physical links, the solution based on the Steiner tree may be sub-optimal, since it does not consider the hierarchical structure of VPN tunnels explicitly. In Cohen and Kaempfer [7], it has been noted that the solution of the optimal tunnel design problem induces a *logical spanning tree* (logical connectivity) rooted at a source node, while this *logical* tree may actually induce a subgraph that is *not a tree* (physical connectivity) in the underlying physical network.

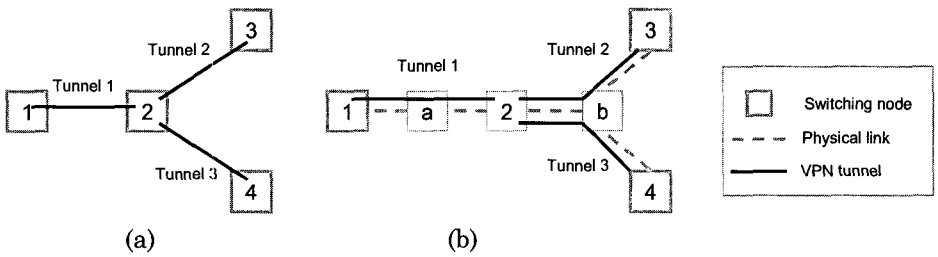

Figure 1. A graphical illustration of the hierarchical routing structure of VPN tunnels in virtual private networks. (a) A *logical* layout of VPN tunnels (b) The associated *physical* layout

Consider the physical graph shown in Figure 2 (a), where it is assumed that all the weights associated with the physical arcs are identical at the value of 1 and the pre-defined limit of VPN tunnel hop count is 2 (i.e., any route of the given traffic demands can have at most 2 VPN tunnels). Figure 2 (c), (d) and (e) show VPN tunnel layouts whose physical paths are determined based on the Steiner tree given in Figure 2 (b), while logical layouts are different from each other. The

layout given in Figure 2 (c) consists of 1-hop VPN tunnels which is set up along the physical path in the Steiner tree. Therefore, the setup overhead is at minimum while the network load is 4 and the maintenance cost of the layout is 12 (=3 +2+3+4). On the contrary, Figure 2 (d) shows an infeasible layout whose cost is at minimum. The network load is 1 and the maintenance cost is 8 (=1+2+1+1+1+2), while the route from node 0 to node 4 is 3-hop path which means that the solution fails to satisfy the hop restriction and it may have a lengthy set-up overhead. Figure 2 (e) shows a modified VPN tunnel layout, which is obtained after concatenating VPN tunnels in the layout given in Figure 2 (d) to decrease the VPN tunnel hop count of the demand from node 0 to node 4. In ths layout, the network load is increased to the value 2 and the maintenance cost is also increased to 9 (=1+2+1+2+3). Unlike the layouts explained above, the layout given in Figure 2 (f) uses physical arcs not in the Steiner tree. The network load is 1 and the maintenance cost is 9 (=1+2+1+2+1+2) while satisfying the hop restrcition. Therefore, this non-Steiner-tree-based VPN tunnel layout is the best feasible solution among the layouts given in Figure 2. This implies that a further investigation is needed to find an optimal solution of the proposed VPNTD problem, which may be obtained by using the layered network flow model given in Section 3.

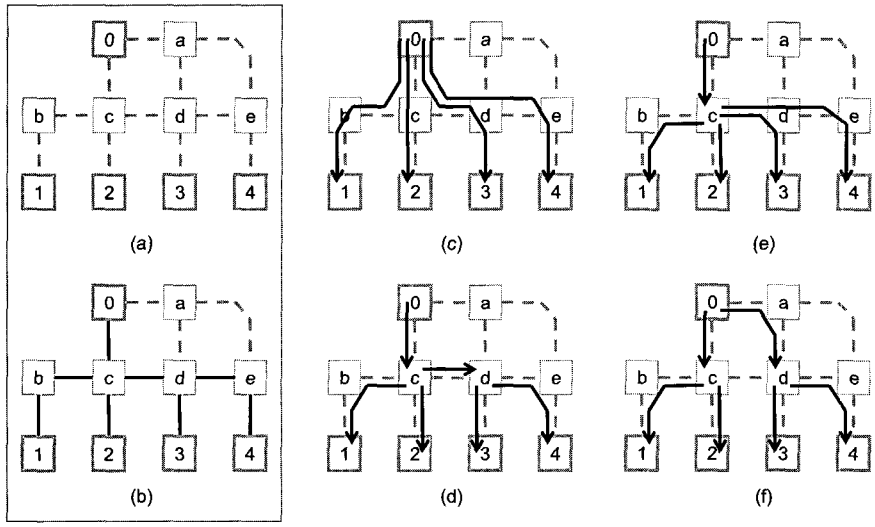

Figure 2. Solutions to the proposed VPNTD problem, where node 0 is the source node and nodes 1, 2, 3 and 4 are the destination nodes. (a) A given physical graph (b) A minimum-cost Steiner tree spanning all the VPN endpoints (c) A tunnel layout with 1-hop VPN tunnels (d) An infeasible VPN tunnel layout based on the Steiner tree (e) A modified tunnel layout based on the Steiner tree (f) A non-Steiner-tree-based solution

Traditionally, this layered structure has been treated in a hierarchical way using well-known multicommodity network flow models. However, *layered network flow models* have been recently devised to consider the connectivity relationship between the logical and physical layers in the integrated model. Dahl, Martin and Stoer [8], Narasimhan, Soni and Song [15] and Saniee and Sokol [16] have proposed some useful algorithms for the layered network flow optimization problem. Especially Sung and Song [5] established a concept of the layered network flow model and proposed path-based branch-and-price algorithm for the virtual path design problem in ATM networks. The proposed layered network flow model considers physical and logical connectivity at the same time and the computational results showed that the solution approach is quite promising. Therefore, the objective of this paper is to investigate the problem of determining a layout of VPN tunnels and to apply the layered network flow approach to the problem. The contribution of this paper includes the successful adaptation of the layered network flow model developed in [5] into the VPN tunnel design problem for which only Steiner tree based heuristics have been proposed previously. The mathematical programming formulation provides the transparency to the VPN tunnel design problem. Some model variants which can be solved efficiently using the proposed algorithm are investigated as well. Based on the branch-and-price algorithm in [5], column generation and branching methods are tailored for the proposed VPNTD problem where demand patterns and objective functions are quite different. Preprocessing methods are devised to deal with the specific demand patterns inherent in the problem – rooted demand trees.

2. VPN TUNNEL DESIGN PROBLEM: DESCRIPTION AND FORMULATION

This section describes mathematically the problem of optimally designing the VPN tunnel layout. Graph representation method used in the paper follows the one introduced in Sung and Song [5]. The network structure of the proposed VPN tunnel design problem consists of a physical graph G=(V, E) and a logical graph N=(V, L). Let T be the set of VPN endpoints. Then, the node set V is decomposed into the terminal node set T and the non-terminal node set R ($V = T \cup R$). It is assumed that one of the VPN endpoints connected by the VPN is the root node where most of the application servers are located and the majority of traffic is sent from the root node to the other VPN endpoints. This means that the terminal node set T is further decomposed into one source node, t_0, and $|T|$-1 destination

nodes, t_q. The non-terminal node set R corresponds to the set of Steiner nodes in the typical Steiner-tree problem (see Koch and Martin [13]). The VPN tunnel hop count of each traffic demand is bounded by the pre-determined parameter, H, which implies that each route becomes a hop-constrained logical path in the logical graph N.

The VPNTD problem can be modeled by means of an arc-flow-based formulation, which has been applied to various layered network flow models including Saniee and Sokol [16] and Narasimhan, Soni and Song [15]. However, it has been noted that a path-based formulation can solve the problem more efficiently and effectively [5]. The path-based formulation proposed in Sung and Song [5] assumed no hop restriction and no demand pattern such as client-server. Since in this paper it is assumed that there exist VPN tunnel hop restriction and the demand pattern is client-server structure, the formulation given in Sung and Song [5] is refined as follows;

q = traffic demand from the source node t_0 to the destination node t_q ($q \in Q$)

H = the pre-defined limit of VPN tunnel hop count

$P_N(q)$ = $\{h \mid h$ is a hop-constrained path for the demand q in the logical graph $N\}$

$P(l)$ = $\{k \mid k$ is a physical path for the logical arc l in the physical graph $G\}$

$L(h)$ = the set of logical arcs, $l \in L$, which compose the hop-constrained logical path h

$E(k)$ = the set of physical arcs, $e \in E$, which compose the physical path k

F = the weight associated with the network load w

m_e = the weight associated with each physical arc e

M_k = the weight associated with the physical path k which indicates the maintenance cost of the associated VPN tunnel l ($M_k = \sum_{e \in E(k)} m_e$, $k \in P(l)$)

z_l = VPN tunnel setup variable for $l=(i, j)$

$$x_{qh} = \begin{cases} 1, \text{if the demand } q \text{ is routed through the hop-constrained logical path } h \\ 0, \text{otherwise} \end{cases}$$

$$y_{lk} = \begin{cases} 1, \text{if the logical arc } l \text{ uses the physical path } k \\ 0, \text{otherwise} \end{cases}$$

w = network load variable (the maximum number of VPN tunnels that share a physical arc in the physical graph)

Without loss of generality, it is assumed that all the coefficients of the objective function are integral. Then, the mathematical formulation of the VPNTD

problem is derived based on the path-based model, called (*HMP*), as follows;

Model (HMP)

$$\textit{Minimize} \quad F \cdot w + \sum_{l \in L} \sum_{k \in P(l)} M_k y_{lk} \tag{1}$$

subject to

$$\sum_{h \in P_N(q)} x_{qh} = 1 \qquad , \forall q = (t_0, t_q) \in Q \tag{2}$$

$$\sum_{\{h \in P_N(q) \mid l \in L(h)\}} x_{qh} \leq z_l \qquad , \forall l = (i, j) \in L, q = (t_0, t_q) \in Q \tag{3}$$

$$\sum_{k \in P(l)} y_{lk} = z_l \qquad , \forall l = (i, j) \in L \tag{4}$$

$$\sum_{l \in L} \sum_{\{k \in P(l) \mid e \in E(k)\}} y_{lk} \leq w \qquad , \forall e = (s, t) \in E \tag{5}$$

$$x_{qh}, y_{lk}, z_l \in B, w \in Z_+ \tag{6}$$

In the model (*HMP*), constraints (2) and (4) mean that one logical path and one physical path are selected for each demand and VPN tunnel, respectively. Constraints (3) are the setup constraints and constraints (5) are the load constraints. Logical path connectivity and physical path connectivity are guaranteed by the sets $P_N(q)$ and $P(l)$, respectively, since their elements are defined as the hop-constrained logical paths and the physical paths, respectively.

3. EXACT SOLUTION ALGORITHM

In the problem description, it is noted that the solutions for the proposed VPNTD problem are similar to those for the directed Steiner tree problem. Therefore, the relationship between the proposed VPNTD problem and the directed Steiner tree problem is investigated in detail. The directed Steiner tree problem is defined as follows; Given a directed simple graph G, a specified source node t_0, and a set of destination nodes $T \subseteq V$, *the directed Steiner tree problem* is the problem of finding the minimum cost arborescence rooted at t_0 and spanning all the nodes in T [6].

***Theorem* 1.** The proposed VPNTD problem without the VPN tunnel hop restriction incorporated is equivalent to the directed Steiner tree problem.

Proof. If the VPN tunnel hop restriction is relaxed, the VPNTD problem becomes the problem of finding a VPN tunnel layout which connecting the source node to the destination nodes, while minimizing both the network load and the mainte-

nance cost of the established VPN tunnels. Therefore, a solution based on the associated minimum-cost directed Steiner tree can be transformed into a valid solution of the relaxed VPNTD problem by decomposing it into paths with endpoints either at the source, destination, or fork nodes of the tree. Since the directed Steiner tree minimizes the established edge costs, each sub-path (corresponding to a VPN tunnel) has a minimum weight which implies that the total maintenance cost of VPN tunnel layout is minimized. Furthermore, the network load remains to be the value 1. Therefore, the Steiner tree becomes an optimal VPN tunnel layout in the relaxed VPNTD problem.
This completes the proof.

Theorem 1 states that the proposed VPNTD problem is the generalization of the directed Steiner tree problem, which has been known to be NP-hard [6].

***Corollary* 1.** The proposed VPNTD problem is NP-hard.

Accordingly, this paper focuses its efforts on deriving an *efficient* branch-and-bound algorithm for the proposed designing issue. In Sung and Song [5], the strength of the path-based LP model is the same as that of the arc-based LP model. However, the path-based model (*HMP*) gives a stronger formulation than the arc-flow-based model (*HAP*) proposed in Appendix A.

***Theorem* 2.** The objective function value of the Linear Programming (LP) relaxation of the path-based model (*HMP*) is greater than or equal to that of the LP relaxation of the arc-flow-based model (*HAP*).

Proof. Let us partition the constraints set of the model (*HAP*) into the set U of (A2), (A3) and (A5) and the set V of (A4) and (A6). The set U is further partitioned into the set $U_1(q)$ of (A2) and (A3) and the set $U_2(l)$ of (A5). It has been shown that the LP relaxation of the model (*HMP*) without the hop-restriction incorporated, called (*MP*), is a Dantzig-Wolfe decomposition of the LP relaxation of the model (*HAP*) without the hop-restriction incorporated, called (*AP*). Therefore, the LP relaxations of the model (*MP*) and (*AP*) are equivalent to each other. A similar approach could be used to the models (*HMP*) and (*HAP*), which considers logical-path-hop restriction. Suppose that the polyhedron $U_1(q)$ has integrality (which implies that any extreme point of $U_1(q)$ is integral). Then each integral extreme point (X_{ij}^q) represents the situation that the demand q is routed through a single logical path $h \in P_N(q)$ and it can be easily shown that the LP relaxation of (*HMP*)

becomes a Dantzig-Wolfe decomposition of the LP relaxation of (*HAP*). However, the convex hull of $U_1(q)$ is not an integral polyhedron, which means that the extreme points of $U_1(q)$ are not integral [2]. Therefore, the formulation of the model (*HAP*) is weaker than that of the model (*HMP*).
This completes the proof.

The result of Theorem 2 implies that the lower bounds derived from the LP relaxations of the model (*HMP*) are better than those from the LP relaxations of the model (*HAP*). Moreover, the problem size of the model (*HMP*) is smaller than that of the model (*HAP*), since the model (*HMP*) has less constraints than the model (*HAP*). And also, the number of columns of the model (*HMP*) may be much smaller than that of the model (*HAP*) because of the employed column generation approach. Therefore, it is expected that the proposed branch-and-bound algorithm with the column generation approach may outperform a standard branch-and-bound algorithm for the model (*HAP*).

3.1 Preprocessing

Preprocessing is a very important algorithmic tool in solving combinatorial and integer programming problems of large scale. The idea in general is to detect unnecessary information in the problem description and to reduce the size of the problem by logical reviews. If the given physical graph is connected (which means that there exists at least one physical path between nodes), the induced logical graph becomes a complete graph having $|V| \cdot |V-1|$ arcs (logical arcs) and it is very burdensome to consider all the logical arcs in the formulation. However, some redundant logical and physical arcs can be removed based on degree tests. Let $\pi^+(v)$ ($\pi^-(v)$) be the number of physical arcs *originated from* (*directed toward*) the node v in the physical graph G. Since the given physical graph is bi-directional, $\pi^+(v) = \pi^-(v) = \pi(v)$. For the proposed VPNTD problem, the following degree test can be implemented before formulating the problem;

1) If $\pi(v) = 1$ for $v \in R \subseteq V$, then the node v and the associated physical and logical arcs can be removed.
2) If $\pi(v) = 2$ for $v \in R \subseteq V$, then the associated logical arcs, which are either originated from or directed toward the node v, can be removed. Furthermore, if two nodes incident to the node v are not connected by any physical arcs, the node v and the associated physical arcs can be removed and a new physical

arc connecting the two nodes incident to the node v is added, whose cost is the sum of the removed physical arcs.

3.2 LP Solution: Column Generation Approach

A natural approach for solving LP relaxations of the model (*HMP*) with an enormous number of columns is to generate columns when they are needed and handle others implicitly. In the proposed algorithm, the model (*HMP*) initially includes all the z-variables, z_l, and the network load variable, w, while the other variables, x_{qh} and y_{lk}, are generated when necessary. Let $\alpha_q, \beta_{ql}, \gamma_l$ and δ_e be the dual variables for constraints (2), (3), (4) and (5), respectively. Also let α_q^*, β_{ql}^*, γ_l^* and δ_e^* be the optimal values of the dual variables of the current model (*HMP*). If the reduced cost of x-variable, $\{-(\alpha_q^* + \sum_{l \in L(h)} \beta_{ql}^*)\}$, turns out to be a negative value, then the associated column can be added, by which the pricing problem for x-variables, x_{qh}, can be formulated as

$$(HSP_1(q)) \qquad \begin{aligned} &min \sum_{l \in L(h)} (-\beta_{ql}^*) - \alpha_q^* \\ &subject\ to \quad h \in P_N(q) \quad \text{and} \quad | L(h) \le H |. \end{aligned}$$

Note that $HSP_1(q)$ is the problem to find the hop-constrained shortest path from node $u(q)$ to node $v(q)$ in the logical graph N, where $u(q)$ is the source and $v(q)$ is the destination of traffic demand q, respectively. Since the arc weights, $(-\beta_{ql}^*)$, are nonnegative, $HSP_1(q)$ can be solved in polynomial time by using a truncated version of the method of successive approximations, whose complexity is of the order $O(H \cdot | V |^2)$ [14]. If the objective value of $HSP_1(q)$ is negative, which indicates that the associated reduced cost is negative, then the logical path will be added to the current formulation of the model (*HMP*) as a new column. Otherwise, no column will be generated with respect to the traffic demand q. For y-variables, y_{lk}, the pricing problem can be formulated in a similar way to $HSP_1(q)$ as

$$(HSP_2(l)) \qquad \begin{aligned} &min \sum_{e \in E(k)} (m_e - \delta_e^*) - \gamma_l^* \\ &subject\ to \quad k \in P(l). \end{aligned}$$

$HSP_2(l)$ is the problem of finding the shortest path from node $u(l)$ to node $v(l)$ in the physical graph G. $HSP_2(l)$ can also be solved very efficiently by Dijkstra's al-

gorithm, whose complexity is of the order $O(|V|^2)$ [4]. If the objective value is negative, then the path will be added to the current formulation of the model (*HMP*) as a new column. Otherwise, no column will be generated with respect to the VPN tunnel l.

3.3 LP Solution: Initial LP Construction

As explained earlier, it is impossible in practice to get all the feasible columns in the beginning. Therefore, the column generation scheme is trying to add columns when necessary and the expectation is that the number of columns necessary to find an optimal solution may not be that much. To help column generation, we need to solve two problems. The first one is how to generate "good" columns quickly and the second one is the initial LP construction. The proposed algorithm first allocates one *logical* path for each origin-demand pair and one *physical* path for each logical path. The logical path is a direct link between two nodes and the physical path for the logical path is the shortest path between the nodes. To guarantee feasibility, dummy variables x_0 and y_0 are included.

$$\textit{Minimize} \quad F \cdot w + \sum_{l \in L} \sum_{k \in P(l)} M_k y_{lk} \tag{1}$$

subject to

$$x_0 + \sum_{h \in P_N(q)} x_{qh} = 1 \qquad , \forall q = (t_0, t_q) \in Q \tag{2}$$

$$x_0 + \sum_{\{h \in P_N(q) \mid l \in L(h)\}} x_{qh} \le z_l \qquad , \forall l = (i, j) \in L, q = (t_0, t_q) \in Q \tag{3}$$

$$y_0 + \sum_{k \in P(l)} y_{lk} = z_l \qquad , \forall l = (i, j) \in L \tag{4}$$

$$y_0 + \sum_{l \in L} \sum_{\{k \in P(l) \mid e \in E(k)\}} y_{lk} \le w \qquad , \forall e = (s,t) \in E \tag{5}$$

$$x_{qh}, y_{lk}, z_l \in B, w \in Z_+ \tag{6}$$

3.4 Exact Algorithm: Branch-and-price Algorithm

Because the LP relaxation of the model (HMP) is solved using the column generation approach, the branch-and-price procedure must allow columns to be generated at each node of the tree to find optimal solutions. Therefore, the key to develop a branch-and-price procedure is to identify a branching rule that eliminates the current fractional solution without compromising the tractability of the pricing (column generation) algorithm. The branching scheme in Sung and Song [5] can be applied to the proposed problem including node selection, fathoming strat-

egy and termination criteria. However, branching variable selection is not apparent so that we further investigate which variables are to be fixed. Let S_{HMP} be the polyhedron from the constraints of the model (*HMP*) (S_{HMP} = {(w, x, y, z) | (w, x, y, z) *satisfies the constraints* (2) – (6)}). Also let $S_{HMP}(z^*, w^*)$ be the polyhedron S_{HMP} obtained after fixing the network load variable, w, and z-variables, z, at integer values, w^* and z^*. Then it is easy to see whether $S_{HMP}(z^*, w^*)$ is an integral polyhedron.

Theorem 3. $S_{HMP}(z^*,w^*)$ is an integral polyhedron.

Proof. It can be proved by showing that the coefficient matrix of $S_{HMP}(z^*, w^*)$ is totally unimodular. If w^* and z^* is fixed at integer values, then the coefficient matrix of the polyhedron $S_{HMP}(z^*, w^*)$ can be partitioned into a matrix A of constraints (2) and (3) and a matrix B of constraints (4) and (5). Since the matrices A and B are the same as the ones introduced in Theorem 2 of Sung and Song [5] and thus TU, the polyhedron is integral.
This completes the proof.

The result of Theorem 3 implies that a branching tree can be designed based only on z-variables, z, and network load variable w: If all the z-variables and network load variable (z, w) are set at integer values in the model (*HMP*), then the associated x-variables and y-variables will attain integer values. This means that branching only on fractional z-variables and network load variable is sufficient to search for all the feasible integer solutions. Accordingly, the size of the branching tree can be reduced greatly. Furthermore, it is noted that, since the branching rule can proceed only with the network load variable and z-variables (which are added initially), the pricing problems, $HSP_1(q)$ and $HSP_2(l)$, will not have to be modified. Branching variables are selected based on the following variable-dichotomy criterion: Given a fractional LP solution, a variable, z_l or w, being at its maximal integer infeasibility (farthest from integrality) is selected as a branching variable, where the integer infeasibility is computed by the function $|0.5 - z_l|$ for z_l and $|0.5 - (w - \lfloor w \rfloor)|$ for w. When there exist ties among the values (measures) of such integer infeasibility, the network load variable w has the higher priority than any other z-variables, since a change in the objective function value with the fixed network load variable may be more drastic than that with z-variables. If only z-variables are in the list, then a logical arc, through which a larger number of traffic demands are routed in the current solution, is selected from among all the associated candidates as a branching variable.

4. COMPUTATIONAL EXPERIMENTS

To evaluate the effectiveness and efficiency of the proposed solution algorithm, it is applied to various numerical test problems. In the experiments, a network generator proposed by Waxman [17] is used to generate random networks with different characteristics. In the Waxman model, nodes are placed randomly in a rectangular coordinate (100×100 grid) by generating uniformly distributed random numbers for each coordinate. Then a directed edge e is placed from $u(e)$ to $v(e)$ at probability $P(u(e),v(e)) = \beta \cdot e^{-d(u(e),v(e))/L \cdot \alpha}$ where $d(u(e), v(e))$ is the Euclidean distance between $u(e)$ and $v(e)$, L is the maximum distance between any two nodes in the graph and α and β are parameters that control the characteristics of the graph generated. Increasing β increases the average vertex degree of the graph and increasing α increases the ratio of longer edges relative to shorter ones. To make the graph connected, a spanning tree is constructed randomly and then some more arcs are added based on the probability $P(u(e), v(e))$. The control provided by this method is allowed for experiments with a wide variety of random graphs. The weight associated with each physical arc is randomly chosen in the range of from 2 to 22. A subset of the nodes in each graph is also chosen randomly as the VPN endpoints. The algorithms are coded in C language and all the tests are made on IBM-PC with a 2.4 GHz Pentium IV processor having 512 Mbytes of RAM. CPLEX 6.5 is used to solve the LP relaxations of the model (*HMP*) in the column generation procedure. The proposed algorithm is then tested with various problem instances which are obtained by varying the number of switching nodes, the number of VPN endpoints, the graph-control parameters α and β, the weight associated with the network load, F, and the maximum number of VPN tunnel hop count, as follows; number of nodes $|V|$ (setting at 20 and 30), number of VPN endpoints $|T|$ (varied from 10 to 27), α (setting at 0.9), β (setting at 0.1, 0.3 and 0.5), F (setting at 30 and 300) and H (setting at 2, 3 and 4).

For comparison, CPLEX is also used to solve the proposed VPNTD problem by using the LP relaxation of the model (*HAP*) together with the branch-and-bound methodology. Regarding the associated branch-and-bound scheme, default settings are used in CPLEX. For instance, the default settings include a general-purpose primal heuristic, a best-bound-first search strategy which generally may give a better performance than any other search orders. The execution time is limited to 7200 seconds for both CPLEX and the proposed solution algorithm.

Table 1. A comparison between the LP relaxations of the model (*HMP*) and (*HAP*) (Test results for the graphs with $|V|=20$ and $F=30$)

			Model (*HAP*) : LP relaxation				Model (*HMP*) : LP relaxation			
$\|T\|$	α,β	H	Ω_{LPR}	*time*	N_{col}	N_{row}	Ω_{LPR}	*time*	N_{col}	N_{row}
18	0.9,0.1	2	190.6	39	35721	14493	198.2	5	1064	6933
		3	203.3	479	33441	14487	207.7	8	1899	6927
		4	183.0	31	34201	14489	183.0	13	2401	6929
	0.9,0.3	2	154.3	1183	63081	14565	163.5	4	1058	7005
		3	123.3	1853	63081	14565	138.3	9	2014	7005
		4	148.0	1707	67641	14577	152.0	24	3624	7017
	0.9,0.5	2	114.6	4172	85881	14625	127.4	4	957	7065
		3	121.7	3473	83601	14619	138.0	8	1769	7059
		4	113.2	2557	96521	14653	121.0	13	2396	7093
14	0.9,0.1	2	267.2	275	28881	12875	271.6	2	855	5395
		3	129.3	513	38761	12901	134.0	3	1296	5421
		4	171.0	66	34201	12889	171.0	11	2502	5409
	0.9,0.3	2	122.6	1519	58521	12953	135.5	1	778	5473
		3	106.4	1405	65361	12971	124.0	8	1603	5491
		4	109.0	1388	63841	12967	109.0	13	2687	5487
	0.9,0.5	2	94.7	2737	86641	13027	101.0	5	983	5547
		3	80.6	2592	84361	13021	90.0	7	1381	5541
		4	108.1	3296	82081	13015	114.0	12	2742	5535
10	0.9,0.1	2	132.0	263	33441	11287	132.0	1	709	3887
		3	103.0	27	29641	11277	105.0	2	1056	3877
		4	137.0	22	28881	11275	137.0	7	1670	3875
	0.9,0.3	2	77.8	1317	63841	11367	87.0	6	959	3967
		3	80.4	1264	61561	11361	89.0	6	1555	3961
		4	99.4	1045	63841	11367	110.5	9	2311	3967
	0.9,0.5	2	66.6	1748	84361	11421	81.0	4	886	4021
		3	60.3	1976	86641	11427	69.0	5	1379	4027
		4	86.0	1465	80561	11411	97.0	11	2386	4011
			125.3	1422.7	60688.4	12958.7	132.8	7.4	1663.7	5478.7

Note: $|V|$ = number of nodes, $|T|$ = number of VPN endpoints, Ω_{LPR} : objective function value of the LP relaxation, N_{col} = number of columns, N_{row} = number of rows

Table 1 shows the comparison result between the LP relaxations of the model (*HMP*) and (*HAP*) for graphs having 20 nodes. The result appears quite impressive, since it took only 7.4 seconds to optimize the LP relaxations of the model (*HMP*), while 1422.7 seconds being taken for the model (*HAP*) by CPLEX. It is noted that, even if the model (*HMP*) has an extremely large number of columns, the average number of columns generated during the column generation procedure is much smaller than the number of columns in the model (*HAP*). Also the number of rows (constraints) of the model (*HMP*) is smaller than those of the model (*HAP*). However, the results for the graphs with 30 nodes are not reported, since CPLEX failed to solve the LP relaxations of those instances within 7200 seconds. However, it took 59.6 seconds to find the optimal solution of the LP relaxation of the model (*HMP*) and the average number of columns generated during the column generation phase is about 3836. The number of constraints is about 18613. Therefore, it is concluded that the LP relaxation of the model (*HMP*) by the column generation methodology may be much more efficient than that of the model (*HAP*) by any standard primal simplex algorithm. Also, the objective function values of the LP relaxations of the model (*HMP*) are larger than those of the model (*HAP*) as expected. Based on the efficiency of the column generation algorithm incorporating tighter lower bounds from the model (*HMP*), it is expected that the proposed exact solution algorithm may outperform the branch-and-bound algorithm for the model (*HAP*) by CPLEX.

Tables 2 and 3 show the detailed test results of the exact solution algorithm for the instances with $|V| = 20$. Both the CPLEX and the proposed solution algorithm solve all the test instances optimally within 7200 seconds. The proposed solution algorithm shows much better performance. Due to the tighter lower bounds from the model (*HMP*), the LP relaxations of the model (*HMP*) found the integer optimal solutions in many test instances. However, the LP relaxations of the model (*HAP*) gave integral (optimal) solutions in just a few instances and therefore the branch-and-bound algorithm for the model (*HAP*) has explored a lot of branching nodes to find integer optimal solutions. Furthermore, for the instances with $F = 30$ (in Table 2), it took only 10.1 seconds for the solution algorithm to find the optimal solutions after exploring 11.9 nodes, while it took 541.4 seconds for CPLEX after exploring 161.4 nodes. Also for the instances with $F = 300$, the proposed solution algorithm found optimal solutions much faster. After analyzing the test results in detail, it can be observed that as the number of VPN endpoints increases, the time to solve the test instances seems to increase. As the hop-limit H increases, the performance of the proposed algorithm becomes worse, since the complexity of the algorithm for pricing subproblem $HSP_1(q)$ depends on

Table 2. Test results for the graphs with $|V| = 20$ and $F = 30$

			Model (*HAP*) : Branch-and-bound					Model (*HMP*) : Branch-and-price				
$\|T\|$	α,β	H	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}
18	0.9,0.1	2	199	282	65	14493	35721	199	5	2	6933	1084
		3	208	342	71	14487	33441	208	10	2	6927	1989
		4	183	5	0	14489	34201	183	13	0	6929	2401
	0.9,0.3	2	179	1318	839	14565	63081	179	26	220	7005	1299
		3	144	5173	1728	14565	63081	144	48	30	7005	3266
		4	152	340	26	14577	67641	152	24	0	7017	3624
	0.9,0.5	2	129	1134	297	14625	85881	129	5	8	7065	977
		3	139	2246	424	14619	83601	139	11	4	7059	1974
		4	121	821	111	14653	96521	121	13	0	7093	2396
14	0.9,0.1	2	292	120	78	12875	28881	292	5	18	5395	1143
		3	134	94	5	12901	38761	134	3	0	5421	1296
		4	171	7	0	12889	34201	171	11	0	5409	2502
	0.9,0.3	2	140	438	187	12953	58521	140	3	30	5473	804
		3	124	458	94	12971	65361	124	8	0	5491	1603
		4	109	9	0	12967	63841	109	13	0	5487	2687
	0.9,0.5	2	101	245	141	13027	86641	101	5	0	5547	983
		3	90	204	18	13021	84361	90	7	0	5541	1381
		4	114	263	13	13015	82081	114	12	0	5535	2742
10	0.9,0.1	2	132	5	0	11287	33441	132	1	0	3887	709
		3	105	15	2	11277	29641	105	2	0	3877	1056
		4	137	3	0	11275	28881	137	7	0	3875	1670
	0.9,0.3	2	87	53	11	11367	63841	87	6	0	3967	959
		3	89	205	20	11361	61561	89	6	0	3961	1555
		4	111	175	35	11367	63841	111	9	2	3967	2329
	0.9,0.5	2	81	141	27	11421	84361	81	5	4	4021	897
		3	69	233	36	11427	86641	69	5	0	4027	1379
		4	97	289	130	11411	80561	97	11	0	4011	2386
			134.7	541.4	161.4	12958.7	60688.4	134.7	10.1	11.9	5478.7	1744.1

Note: Ω_{OPT} : optimal objective function value after terminating branching procedures, N_{Br} = number of branching nodes explored, N_{col} = number of columns, N_{row} = number of rows

Table 3. Test results for the graphs with $|V| = 20$ and $F = 300$

			Model (*HAP*) : Branch-and-bound					Model (*HMP*) : Branch-and-price				
$\|T\|$	a,β	H	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}
18	0.9,0.1	2	512	1659	787	14495	36481	512	18	8	6933	1602
		3	457	1398	698	14505	40281	457	19	14	6945	2402
		4	447	51	2	14503	39521	447	21	0	6943	3503
	0.9,0.3	2	430	1547	354	14551	57761	430	3	14	6991	1005
		3	426	1338	313	14553	58521	426	16	12	6993	2069
		4	408	191	24	14559	60801	408	12	2	6999	2412
	0.9,0.5	2	391	5381	248	14637	90441	391	7	8	7077	1136
		3	442	7202	538	14605	78281	401	10	0	7045	2018
		4	397	998	70	14607	79041	397	29	0	7047	4204
14	0.9,0.1	2	480	260	34	12903	39521	480	5	2	5423	1047
		3	412	108	23	12887	33441	412	6	0	5407	1948
		4	418	181	32	12899	38001	418	23	2	5419	3986
	0.9,0.3	2	398	161	17	12953	58521	398	4	4	5473	919
		3	378	109	12	12975	66881	378	4	0	5495	1364
		4	379	98	0	12971	65361	379	11	0	5491	2598
	0.9,0.5	2	367	445	18	13035	89681	367	2	0	5555	861
		3	386	7201	690	13003	77521	381	4	2	5523	1429
		4	380	2427	140	13015	82081	380	14	0	5535	3102
10	0.9,0.1	2	431	16	2	11289	34201	431	3	0	3889	878
		3	395	81	3	11283	31921	395	3	0	3883	1411
		4	384	26	0	11291	34961	384	9	0	3891	2530
	0.9,0.3	2	361	61	4	11355	59281	361	3	0	3955	830
		3	372	127	5	11367	63841	372	3	0	3967	1248
		4	373	109	7	11365	63081	373	13	2	3965	2845
	0.9,0.5	2	357	327	13	11419	83601	357	2	6	4019	751
		3	352	73	0	11417	82841	352	2	0	4017	1015
		4	361	118	21	11423	85121	361	16	6	4023	2850
			403.5	1173.8	150.2	12958.0	60406.9	401.8	9.7	3.0	5477.9	1924.6

Table 4. Test results for the graphs with $|V| = 30$ and $F = 30$

			Model (*HAP*) : Branch-and-bound					Model (*HMP*) : Branch-and-price				
$\|T\|$	α,β	H	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}
27	0.9,0.1	2	*382	7204	437	49678	155731	372	35	26	23668	2149
		3	*258	7204	136	49698	173131	231	150	32	23688	4819
		4	239	4640	59	49676	153991	239	81	0	23666	5158
	0.9,0.3	2	*234	7206	244	49824	282751	186	52	72	23814	2229
		3	*252	7207	24	49874	326251	183	91	2	23864	4605
		4	*191	7206	38	49840	296671	184	188	2	23830	6633
	0.9,0.5	2	*176	7210	124	50026	458491	158	18	6	24016	1950
		3	*156	7209	41	49978	416731	139	68	2	23968	4009
		4	154	6403	35	50016	449791	154	179	0	24006	6786
21	0.9,0.1	2	252	1710	74	44284	160951	252	12	6	18454	1807
		3	*224	7203	164	44294	169651	203	49	0	18464	3989
		4	221	93	0	44254	134851	221	103	0	18424	6194
	0.9,0.3	2	158	3483	267	44472	324511	158	15	10	18642	1844
		3	*174	7206	45	44412	272311	152	85	6	18582	4672
		4	*173	7206	57	44460	314071	160	121	0	18630	6873
	0.9,0.5	2	120	5187	292	44606	441091	120	12	2	18776	1736
		3	119	1635	6	44634	465451	119	41	0	18804	3297
		4	140	2450	9	44622	455011	140	131	0	18792	6255
15	0.9,0.1	2	207	4144	588	38898	173131	207	50	140	13248	2052
		3	148	437	7	38876	153991	148	20	0	13226	2826
		4	174	1254	26	38878	155731	174	105	4	13228	5986
	0.9,0.3	2	126	2736	158	39026	284491	126	35	8	13376	2068
		3	127	3379	68	39058	312331	127	38	2	13408	3332
		4	126	931	19	39074	326251	126	140	2	13424	6479
	0.9,0.5	2	98	1666	60	39166	406291	98	11	2	13516	1511
		3	104	4898	70	39180	418471	104	57	6	13530	3785
		4	120	3230	34	39154	395851	120	80	0	13504	5284
			179.7	4456.9	114.1	44442.9	299184.3	170.4	72.9	12.2	18612.9	4012.1

Note: * Instances in which the branch-and-bound algorithm fails to find optimal solutions (and therefore, the best integer solutions found are reported).

Table 5. Test results for the graphs with $|V| = 30$ and $F = 300$

			Model (*HAP*) : Branch-and-bound					Model (*HMP*) : Branch-and-price				
$\|T\|$	α,β	H	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}	Ω_{OPT}	*time*	N_{Br}	N_{row}	N_{col}
27	0.9,0.1	2	*643	7204	158	49672	150511	580	150	25	23696	8086
		3	*620	7204	69	49686	162691	515	278	20	23676	7397
		4		7335		49676	153991	540	361	0	23666	13744
	0.9,0.3	2	*499	7207	126	49876	327991	459	844	1010	23866	2785
		3		7207		49860	314071	431	55	0	23850	3767
		4		7211		49876	327991	462	304	0	23866	10373
	0.9,0.5	2	*503	7209	50	49984	421951	439	132	136	23974	2369
		3		7209		50020	453271	420	45	2	24010	3343
		4		7209		50008	442831	430	164	0	23998	7565
21	0.9,0.1	2	*498	7203	348	44276	153991	486	17	16	18446	1912
		3	*477	7203	72	44268	147031	468	304	46	18438	7558
		4	487	3159	45	44286	162691	487	78	0	18456	5249
	0.9,0.3	2	*429	7206	243	44452	307111	428	19	30	18622	1805
		3	*427	7207	48	44482	333211	401	37	2	18652	3111
		4		7206		44416	275791	430	142	0	18586	7892
	0.9,0.5	2	389	4945	60	44638	468931	389	17	16	18808	1863
		3		7209		44630	461971	391	37	0	18800	3197
		4	*415	7209	8	44606	441091	408	201	0	18776	8085
15	0.9,0.1	2	449	726	23	38862	141811	449	41	2	13212	2256
		3	426	1115	15	38876	153991	426	31	0	13226	3738
		4	440	1822	23	38868	147031	440	69	0	13218	5704
	0.9,0.3	2	383	7131	55	39058	312331	383	15	0	13408	1613
		3		7207		39078	329731	388	40	6	13428	3385
		4	*391	7206	20	39058	312331	386	92	0	13408	6084
	0.9,0.5	2	369	3982	13	39214	448051	369	32	2	13564	1858
		3	376	6342	23	39208	442831	376	27	0	13558	2776
		4	379	4154	12	39200	435871	379	115	0	13550	5355
			452.6	6045.4	74.3	44449.4	304855.4	435.6	135.1	48.6	18620.7	4921.1

Note: 1. * Instances in which the branch-and-bound algorithm for the model (*HAP*) fails to find optimal solutions (and therefore, the best integer solutions found are reported instead).

2. Blank cells imply that the branch-and-bound algorithm for the model (*HAP*) fails to find even feasible solutions.

the hop-limit H (the complexity is of the order $o(H \cdot |V|^2)$) and therefore it takes more time to solve the pricing subproblem. However, the performance of the proposed algorithm does not seem to be dependent on the network structures which are obtained by varying the control parameters (α, β).

Similar trends can be seen from the instances with $|V| = 30$, which are shown in Tables 4 and 5. Optimal solutions were found within 2 minutes in average by using the proposed solution algorithm for the model (*HMP*), while the branch-and-bound algorithm for the model (*HAP*) failed to solve the problem optimally within two hours in many test instances. In some instances, the branch-and-bound algorithm failed to find even a feasible integer solution within 2 hours. The performance of the proposed solution algorithm seems to depend on the number of VPN endpoints, $|T|$ and the hop-limit, H, which is similar to the performance shown in the test instances with 20 nodes. It seems to be robust to the network structures.

In conclusion, the above extensive computational experiments show that the proposed solution algorithm works quite well in finding optimal solutions within reasonable time. It is also demonstrated that the proposed solution algorithm outperforms the MIP solver (CPLEX 6.5) and is very efficient and effective.

5. MODEL VARIANT AND EXTENSION

In the preceding sections, a basic version of the VPN tunnel design problem which minimizes the cost of operation and maintenance of virtual topologies is investigated in detail. This section briefly describes how to extend the path-based model (*HMP*) to solve the VPN tunnel design problem whose objective is to minimize the number of active switching nodes, where a switching node is defined to be *active* if the node is one of the endpoints of any logical tunnels. By definition, any VPN endpoints (i.e., the source and destination nodes) are active.

5.1 VPN Tunnel Layout with Minimal Active Node Set

The problem, called *Minimal Active Set VPN* (*MASVPN*) problem, is defined as follows; Given a directed simple graph $G(V, E)$, a set of VPN end points $T \subseteq V$ and a source node $t_0 \in T$, find an arborescence (directed tree) rooted at t_0 and spanning the other VPN end points with a set of fork nodes A such that $|A \backslash T|$ is minimal. Since the MASVPN problem is NP-hard, a heuristic based on the directed Steiner tree problem is proposed in Cohen and Kaempfer [7]. The path-

based layered network flow model (*HMP*) can be extended to provide lower bounds and optimal solutions of the MASVPN problem. Let r_i be the indicator variable, which has a value 1 if the node i is active and a value 0 otherwise. Then the formulation of the MASVPN problem is derived based on the path-based model (*HMP*) as follows;

Model (MPMAS)

Minimize $\sum_{i \in V \setminus T} r_i$ (7)

subject to

$\sum_{h \in P_N(q)} x_{qh} = 1$, $\forall q = (t_0, t_q) \in Q$ (8)

$\sum_{\{h \in P_N(q) \mid l \in L(h)\}} x_{qh} \le z_l$, $\forall l = (i, j) \in L, q = (t_0, t_q) \in Q$ (9)

$\sum_{k \in P(l)} y_{lk} = z_l$, $\forall l = (i, j) \in L$ (10)

$\sum_{l \in L} \sum_{\{k \in P(l) \mid e \in E(k)\}} y_{lk} \le 1$, $\forall e = (s, t) \in E$ (11)

$z_l \le r_i$, $\forall l = (i, j) \in L$ (12)

$z_l \le r_j$, $\forall l = (i, j) \in L$ (13)

$x_{qh}, y_{lk}, z_l, r_i \in B$ (14)

It is noted that since every VPN endpoints are active by definition, $r_i = 1$, $\forall i \in T$ and the network load is set at the value 1. The objective function value of the LP relaxations of the model (*MPMAS*) becomes a lower bound for the MASVPN problem. The LP relaxation of the model (*MPMAS*) can be solved by using the column generation approach, which is similar to the approach employed in Section 3. Since the VPN tunnel hop count is not restricted in the MASVPN problem, both the logical and physical paths can be obtained efficiently by using the shortest path algorithm when generating columns associated with x-variables and y-variables. Furthermore, if all the z-variables are set at integer values, then the resulting polyhedron obtained from the constraints of the model (*MPMAS*) becomes an integral polyhedron which implies that branch-and-price algorithm proposed in Section 3 can be applied to the model (*MASVPN*) without modification.

6. CONCLUSION AND FURTHER RESEARCH DIRECTIONS

In this paper, the mathematical formulation and the exact solution algorithm for

the VPN tunnel design problem are presented. The relationship between the proposed VPNTD problem and the minimum-cost directed Steiner tree problem is investigated and a branch-and-price algorithm is developed to solve the proposed problem optimally. Computational results show that the proposed algorithm can solve the proposed problem optimally within a reasonable time. A subject for further study may be a VPN tunnel design problem which considers queueing delay at intermediate switching nodes as a quality-of-service measure for customers. Also, it may be useful to analyze polyhedral characteristics of the proposed VPNTD problem to strengthen the mathematical formulation. Theorem 4 shows the possibility of strengthening the lower bounds from the model (*HMP*) further.

***Theorem* 4.** An optimal solution of the proposed VPNTD problem induces an arborescence rooted at the source node t_0 in the logical graph N.

Proof. Suppose that there exists an optimal solution that does not induce an arborescence (a directed tree rooted at the source node t_0) in the logical graph N, which means that there exist multiple logical paths from the source node t_0 to some destination node t_q. Then, even if some redundant logical paths are deleted, the solution composed of the remaining logical paths is still valid, while the objective function value may decrease. Therefore, it cannot be an optimal solution. This completes the proof.

Thereupon, some valid inequalities, which have been proposed for the Steiner tree problem, can be applied to the proposed VPNTD problem. This implies that by adding any violated valid inequalities when necessary (*row generation approach*), the lower bounds from the model (*HMP*) may be tightened further.

REFERENCES

[1] Aneroussis, N. and A. A. Lazar, "Virtual path control for ATM networks with call level quality of service guarantees," *IEEE/ACM Transactions on Networking* 6, 2 (1998), 222-236.

[2] Balakrishnan, A. and K. Altinkemer, "Using a hop-constrained model to generate alternative communications network design," *ORSA Journal on Computing* 4 (1992), 192-205.

[3] Barnhart, C. et al., "Branch-and-price: Column generation for solving huge

integer programs," *Operations Research* 46, 3 (1998), 316-329.

[4] Bondy, J. A. and U. S. R. Murty, *Graph theory with applications*. Elsevier: New York, 1976.

[5] Sung, C. S. and S. H. Song, "Branch-and-price algorithm for a combined problem of virtual path establishment and traffic packet routing in a layered communication network," *Journal of the Operational Research Society* 54 (2003), 72-82.

[6] Charikar, M. et al., Approximation algorithms for directed Steiner problems. Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998.

[7] Cohen, R. and G. Kaempfer, "On the cost of virtual private networks," *IEEE/ACM Transactions on Networking* 8, 6 (2000), 775-784.

[8] Dahl, G., A. Martin, and M. Stoer, "Routing through virtual paths in layered telecommunications networks," *Operations Research* 47, 5 (1999), 693-702.

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

[10] Duffield, N. G. et al., "Resource management with hoses: Point-to-cloud services for virtual private networks," *IEEE/ACM Transactions on Networking* 10, 5 (2002), 679-692.

[11] Gertstel, O., I. Cidon, and S. Zaks, "Efficient support for client/server applications over heterogeneous ATM networks," *IEEE/ACM Transactions on Networking* 6, 4 (1998), 432-446.

[12] Gerstel, O., A. Wool, and S. Zaks, "Optimal layouts on a chain ATM network," *Discrete Applied Mathematics* 83 (1998), 157-178.

[13] Koch, T. and A. Martin, "Solving Steiner tree problems in graphs to optimality," *Networks* 32 (1998), 207-232.

[14] Lawler, E. L., *Combinatorial optimization: Networks and matroids*, Rinehart & Winston: New York, 1976.

[15] Narasimhan, S., S. Soni, and S. H. Song, "ATM network design for corporate networks," *European Journal of Operational Research* 170 (2006), 644-663

[16] Saniee, I. and J. S. Sokol, "Virtual path design in service-specific ATM networks," *Journal of Heuristics* 6, 1 (2000), 65-83.

[17] Waxman, B. M., "Routing of multipoint connections," *IEEE Journal of Selected Areas in Communication* 6 (1988), 1617-1622.

[18] Hota, C., S. KumarJha, and G. Raghurama, "Restoration of Virtual Private Networks with QoS Guarantees in the Pipe Model," *Lecture Notes in Com-*

puter Science 3326 (2004), 289-302.

[19] Yang, Y., C. U. Martel, and S. F. Wu, "On building the minimum number of tunnels: an ordered-split approach to manage IPSec/VPN policies," *Proceedings of IEEE/IFIP Conference on Network Operations and Management Symposium* 1, 19-23 (2004), 277-290.

[20] Saad, T. et al., "Tunneling Techniques for End-to-End VPNs: Generic Deployment in an Optical Testbed Environment," *IEEE Communications Magazine*, (2006), 124-132.

[21] Chen, I.-W., Y.-D. Lin, and Y.-N. Lin, "Tunnel Minimization and Relay for Managing Virtual Private Networks," *Proceedings of IEEE Conference on Globecom*, (2004), 2128-2133.

APPENDIX

A. An arc-flow-based formulation for the VPNTD problem:

The associated parameters and decision variables are defined as follows;

q = a traffic demand from the source node t_0 to the destination node t_q ($q \in Q$)

H = the pre-defined limit of VPN tunnel hop count

F = the weight associated with the network load W

m_{st} = the weight associated with each physical arc $e=(s, t)$

W = network load variable (the maximum number of VPN tunnels that share a physical arc in the physical graph)

$$Z_{ij} = \begin{cases} 1 & \text{if VPN tunnel } l = (i, j) \text{ is established} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ij}^{q} = \begin{cases} 1 & \text{if traffic demand } q = (t_0, t_q) \text{ is routed through VPN tunnel } l = (i, j) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ij}^{st} = \begin{cases} 1 & \text{if the VPN tunnel } l = (i, j) \text{ uses physical link } e = (s, t) \\ 0 & \text{otherwise} \end{cases}$$

Now an arc-flow-based mathematical formulation for the VPNTD problem, called model (*HAP*), is derived as follows:

Model (HAP)

$$\text{Minimize} \quad F \cdot W + \sum_{(i,j)\in L} \sum_{(s,t)\in E} m_{st} \cdot Y_{ij}^{st} \tag{A1}$$

subject to

$$\sum_{j\in V} X_{ij}^{q} - \sum_{j\in V} X_{ji}^{q} = \begin{cases} 1 & \text{if } i = t_0 \\ -1 & \text{if } i = t_q \\ 0 & \text{if } i \neq t_0, t_q \end{cases} \quad , \forall q = (t_0, t_q) \in Q \tag{A2}$$

$$\sum_{(i,j)\in L} X_{ij}^{q} \leq H \quad , \forall q = (t_0, t_q) \in Q \tag{A3}$$

$$X_{ij}^{q} \leq Z_{ij} \quad , \forall l = (i, j) \in L, q = (t_0, t_q) \in Q \tag{A4}$$

$$\sum_{t\in V} Y_{ij}^{st} - \sum_{t\in V} Y_{ij}^{ts} = \begin{cases} Z_{ij} & \text{if } i = s \\ -Z_{ij} & \text{if } j = s \\ 0 & \text{otherwise} \end{cases} \quad , \forall l = (i, j) \in L \tag{A5}$$

$$\sum_{(i,j)\in L} Y_{ij}^{st} \le W \qquad , \forall e = (s,t) \in E \tag{A6}$$

$$X_{ij}^{q}, Y_{ij}^{st}, Z_{ij} \in B, W \in Z_{+} \tag{A7}$$

The objective function (A1) minimizes the cost of operation and maintenance of the virtual topology (VPN tunnel layout), where the first term represents the cost of network load (the maximum number of VPN tunnels that share a physical arc) and the second term represents the maintenance cost of each VPN tunnel. Constraints (A2) guarantee the logical arc connectivity for each traffic demand and constraints (A5) guarantee the physical arc connectivity for each logical arc. Constraints (A3) represent the VPN tunnel hop count restriction. Constraints (A4) are the conditions that a logical arc (VPN tunnel) can be used by any traffic demands if and only if the logical arc is established. Constraints (A6) are the load constraints. Constraints (A7) are the integrality requirement on the decision variables.