Mesh Topology Design in Overlay Virtual Private Networks

Ezhan Karaşan* ‡ Oya Ekin-Karaşan† Nail Akar* Mustafa Ç. Pınar†

* Department of Electrical and Electronics Eng.

Bilkent University, Ankara, Turkey

† Department of Industrial Eng.

Bilkent University, Ankara, Turkey

‡ Corresponding author (e-mail: ezhan@ee.bilkent.edu.tr)

April 10, 2002

Mesh Topology Design in Overlay Virtual Private Networks

Ezhan Karasan Oya Ekin-Karasan Nail Akar Mustafa C. Pinar

We study the "mesh topology design problem" in overlay virtual private networks. Given a set of customer nodes and an associated traffic matrix, tunnels that are connecting node pairs through a service provider network are determined such that the total multi-hopped traffic is minimized. A tabu search (TS) based heuristic is proposed.

Introduction: The term "Virtual Private Network" (VPN) refers to the communication between a set of nodes of a Customer Network (CN), making use of a shared network infrastructure of a Service Provider (SP). An overlay VPN refers to an approach in which the knowledge of the CN is limited to only the customer edge devices, or equivalently, the SP infrastructure is hidden from the CN. In overlay VPNs, the SP enables the set up of tunnels or virtual links between two customer edge devices. In the case of an IP or ATM service provider infrastructure, tunnels are created by encapsulating packets within another encapsulating header so that the traffic between those two entities can seamlessly be carried through the SP infrastructure. In the case of an infrastructure with optical cross-connects, lightpaths take on the role of tunneling between two electronic entities.

In this paper, we study the "mesh topology design problem" in overlay VPNs. The design starts with a set of customer nodes and an associated traffic matrix, and determines which node pairs are to be connected via tunnels through the SP network. We note that it may not be possible to set up a tunnel between each pair of nodes because a node can be the source and sink of only a limited number of tunnels due to hardware and/or software constraints. In the partial mesh case, traffic between node pairs that are not directly connected must use a sequence of tunnels through intermediate nodes. Typically, this sequence is governed by a shortest path algorithm controlled by the CN. Such multi-hopping has a detrimental cost impact on the CN because of two reasons. First, the SP may charge the CN in proportion with the overall traffic the CN injects into the SP. Secondly, increased multi-hopping increases packet processing requirements in the intermediate nodes. Therefore, we formulate the mesh topology design problem in overlay VPNs in terms of minimizing the overall amount of multi-hopped traffic.

In this paper, we offer an Integer Programming formulation of the problem which is outside the capabilities of off-the-shelf software for even small problem sizes. We resort to TS-based heuristics and lower bounding techniques to bracket the optimum value when no exact optimum can be obtained. We finally discuss our computational experience.

Mathematical Model: Given an $n \times n$ demand matrix $T = [t_{ij}]$ and an integer $p > 0$, we are interested in constructing a simple undirected graph $G = (V, E)$ with node set $V = \{1, \ldots, n\}$ where each node has degree less than or equal to p and routing t_{ij} units of flow from i to j, for all $1 \leq i \neq j \leq n$ so as to minimize the cost defined as the total multi-hopped traffic. A similar problem akin to ours arising in light-wave networks is discussed in [1]. Although the context is different from ours it turns out that their formulation is largely applicable in our case with the main difference being that they minimize the maximum edge flow.

We work with the following integer linear programming formulation of the problem. For each pair of nodes i, j such that $i < j$ we have a $\{0,1\}$-variable a_{ij}, set to 1 if edge $(i, j) \in E$, and set to 0 otherwise. For each pair of nodes k, l and ordered pair i, j we have a binary variable b_{ij}^{kl} that is set to 1 if edge (i, j) is used to carry traffic (in the direction from i to j) between nodes k and l. The formulation is as follows:

$$\min \sum_{ijkl} b_{ij}^{kl} t_{kl}$$

subject to

$$b_{ij}^{kl} + b_{ji}^{kl} \leq a_{ij} \; \forall k \neq l, i < j$$

$$\sum_j b_{ij}^{kl} - \sum_j b_{ji}^{kl} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k, l \end{cases} \quad \forall i, k \neq l$$

$$\sum_{j>i} a_{ij} + \sum_{j<i} a_{ji} = p \; \forall i$$

As the linear programming (LP) relaxation of the above formulation tends to give rather weak lower bounds on the optimal value for small values of p, we strengthen the formulation using flux and distance inequalities.

Let us denote $f(c)$ an aggregate flow quantity defined as follows:

$$f(c) = \sum_{ijl} b_{ij}^{cl} t_{cl}$$

For $d \geq 1$ we define $T^d(c)$ as the sum of the $p^{d-1}+1$ through $p^{d-1}+(p-1)^{d-1}p$ largest demands with source c, and $V^d(c)$ as the set of nodes i such that t_{ci} is included in $T^d(c)$. Let

$$s^d(c) = \sum_{2 \leq h < d} \min\{t_{ci} : i \in V^h(c)\}$$

For convenience, let $\overline{a_{ci}} = a_{lm}$ where $l = \min(c,i)$ and $m = \max(c,i)$. Then, the flux inequality is given by [1]:

$$f(c) \geq \sum_{i \neq c} (2 - \overline{a_{ci}}) t_{ci} + \sum_{d>2} (d-2) T^d(c) + \sum_{d>2} \sum_{i \in V^d(c)} (s^d(c) - (d-2) t_{ci})(\overline{a_{ci}})$$

In addition to the flux inequalities we devise additional valid inequalities that we refer to as distance inequalities. Distance between nodes k and l, say $\delta(k,l)$, is defined as the number of tunnels used in carrying the traffic from node k to node l. Obviously, in any optimal solution $\delta(k,l) = \delta(l,k)$ since the traffic between any pair of nodes will be routed through the shortest path.

A first set of inequalities is obtained from the triangle inequality:

$$\delta(i,j) \leq \delta(i,k) + \delta(k,j) \ \forall i,j,k; \quad \delta(i,i) = 0 \ \forall i; \quad \delta(i,j) \geq 1 \ \forall i \neq j$$

A second set of inequalities is derived by considering an ideal network design for each node c such that the total distance from c to each of the remaining $n-1$ nodes is minimized. In the ideal design, the number of nodes connected to c with a distance of d is at most $(p-1)^{d-1}p$. These considerations lead to the following valid inequality for any $S \subseteq V$:

$$\sum_{i \notin S} \delta(c,i) \geq \sum_{d=1}^{D-1} dp(p-1)^{d-1} + D[n-1-\sum_{d=1}^{D-1} p(p-1)^{d-1}] - D|S| \ \forall c$$

where D is the longest distance in the ideal design (a tree rooted at node c connected to p nodes with distance 1, these p nodes connected each to $p-1$ other nodes and so on ...) between node c and any other node, e.g., $D = 3$ for $n = 20$ and $p = 3, 4$.

TS Heuristic: Our heuristic procedure can be considered as a generic TS algorithm [2]. It starts from an initial feasible solution, i.e., a connected graph where each node has degree p. At each iteration, the method considers all disjoint pairs of edges (i_1, j_1) and (i_2, j_2) and possible edge exchanges so as to preserve the degree of all four nodes involved. Associated with each possible exchange is a cost value which is computed after finding all-pairs shortest paths in the graph resulting from this exchange. The link exchange with the least cost is implemented and this particular exchange is made tabu for a random number of iterations to prohibit cycling. The algorithm stops if the overall best objective value does not improve for T iterations. At the start of the algorithm no edge exchange is tabu. We use two alternative procedures to get an initial feasible solution: 1) we randomly generate 100 feasible solutions and run the heuristic starting from each, recording the best solution obtained, 2) we use the heuristic logical topology design algorithm (HLDA) which generates a feasible topology by assigning edges between node pairs in the order of descending traffic between them [3].

Consider a node quadruple $\{i_1, i_2, j_1, j_2\}$ with respect to the current feasible graph $G = (V, E)$ for which $E_0 = \{(i_1, j_1), (i_2, j_2)\} \subset E$. Let us define the following sets of edges:

$$E_1 = \{(i_1, i_2), (j_1, j_2)\}, \text{ and } E_2 = \{(i_1, j_2), (i_2, j_1)\}$$

We call the node quadruple $\{(i_1, j_1), (i_2, j_2)\}$ feasible if $E_1 \cap E = \emptyset$ or $E_2 \cap E = \emptyset$. We define an exchange graph $G_{(i_1,j_1),(i_2,j_2)} = (V, E \setminus E_0 \cup E')$ with respect to a feasible node quadruple $\{(i_1, j_1), (i_2, j_2)\}$ where $E' = E_1$ if $E_1 \cap E = \emptyset$ and $E_2 \cap E \neq \emptyset$, $E' = E_2$ if $E_1 \cap E \neq \emptyset$ and $E_2 \cap E = \emptyset$, and $E' = E_1$ or E_2 (each with probability 1/2), otherwise. The neighborhood of G, $\mathcal{N}(G)$ is then defined as the union of all exchange graphs corresponding to feasible node quadruples which are not in the tabu list.

Let $cost(G)$ be the objective function value corresponding to the current feasible graph G

obtained by carrying the traffic along the shortest paths in G. A typical iteration runs as follows: Select $G' = argmin_{\bar{G}\in\mathcal{N}(G)} cost(\bar{G})$, and declare the exchange sequence (E_0, E') as tabu for a random number of iterations selected uniformly in the interval $[l, u]$.

Results and Conclusion: We report our computational results with some randomly generated test problems. Our test problems consist of graphs with 20 nodes for $p = 3$ and $p = 4$. For each value of p we generate six problems and set $T = 3000$, $l = 30$ and $u = 100$. Tables 1 and 2 list our computational results. All LPs are solved using the barrier optimization method of CPLEX 7.1. The column LP gives the value of the LP relaxation, the column LP+f the value of the LP relaxation after the addition of flux inequalities, the column LP+fd the value of the LP relaxation after the addition of both flux and distance inequalities. We confined our experiments to the cases where $|S| \leq 3$ for keeping the size of the linear program manageable. Column CH contains the objective value obtained using HLDA in [3]. Columns TSc and TSr contain the objective function values obtained from our heuristic with an initial solution obtained by using HLDA and by selecting the best out of 100 randomly generated initial graphs, respectively. Finally, the last column reports the percent gap between the best lower and upper bounds.

The TS-based heuristic for topology design in overlay VPNs achieves cost values within 2.5% of the optimum for the 20-node network. The solution gets slightly closer to the optimum as p increases. When HLDA is used as the initial graph, our heuristic generates solutions that are within 1% of the best solution obtained when our algorithm is initialized with 100 randomly generated graphs. The computation time of the algorithm is in the order of few minutes for $n = 20$ on a Pentium III PC.

References

[1] D. Bienstock and O. Günlük: 'Computational Experience with a Difficult Mixed-Integer Multicommodity Flow Problem', Mathematical Programming, 1995, 68, 213-238.

[2] F. Glover and M. Laguna: 'Tabu Search' (Kluwer Academic Publishers, Boston, MA, 1997)

[3] R. Ramaswami and K. Sivarajan: 'Design of Logical Topologies for Wavelength-Routed Optical Networks', IEEE Journal on Selected Areas in Communications, 1996, 14, (5), 840-851.

Authors' affiliations:

E. Karasan and N. Akar (Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06533, Turkey)

O. Ekin-Karasan and M.C. Pinar (Department of Industrial Engineering, Bilkent University, Ankara 06533, Turkey)

e-mail: ezhan@ee.bilkent.edu.tr

Table captions:

Table 1. Results with $n = 20$ and $p = 3$

Table 2. Results with $n = 20$ and $p = 4$

Table 1

LP	LP+f	LP+fd	CH	TSc	TSr	GAP
39582.79	47496.95	47891.74	54153	49092	49041	2.40
38316.18	45957.36	46528.82	53408	47817	47585	2.27
37328.07	44939.56	45389.21	54824	46454	46342	2.10
34946.23	44224.82	44702.80	53566	45969	45818	2.50
37797.97	45129.83	45494.31	55230	46658	46613	2.46
34265.00	41039.00	41602.47	48541	42597	42534	2.24

Table 2

LP	LP+f	LP+fd	CH	TSc	TSr	GAP
37842.37	39151.00	39415.99	43781	40481	40238	2.09
36644.99	37837.96	38155.10	44251	39418	38997	2.21
35716.87	37093.46	37409.31	42284	38363	38155	1.99
35293.84	36458.46	36831.70	41379	37716	37628	2.16
36113.88	37224.93	37525.39	42972	38474	38361	2.23
32743.63	33902.52	34226.71	37981	35105	35010	2.29