On-line Routing for Virtual Private Networks (Draft: Do not distribute)

Rainer Gawlick[1] Charles Kalmanek[2] K. G. Ramakrishnan[2]

[1] Laboratory for Computer Science, MIT
[2] AT&T Bell Laboratories

February 2, 1994

Abstract

Today, large telecommunications customers are increasingly using virtual private networks to link geographically dispersed sites. A virtual private network is provided by provisioning a set of permanent (long term) virtual circuits between customer endpoints on a large backbone network. This paper considers the problem of routing a set of permanent virtual circuit requests over a backbone network. Several factors make this routing problem complicated. Routing decisions must be made *on-line* without any knowledge of future request sets. Furthermore, frequent rerouting to correct inefficiencies that can result from the on-line routing decisions are not possible since rerouting creates a service disruption for the customer. Finally, the forward and reverse bandwidth of a virtual circuit must be routed over the same single path.

Using an extensive set of simulations, this paper evaluates several different strategies for on-line permanent virtual circuit routing. We find that a strategy based on recent results in competitive analysis and ideas from combinatorial optimization consistently provides the best performance.

The problem of admission control is closely related to the problem of routing. This paper also provides a theoretical lower bound which suggests that non-greedy admission control is a fundamental component of an efficient on-line permanent virtual circuit routing algorithm.

1. Introduction

Today, large telecommunications customers are increasingly using virtual private networks to link geographically dispersed sites. A virtual private network is provided by provisioning a set of permanent (long term) virtual circuits between customer endpoints on a large backbone network. For example, a customer wishing to connect sites A, B, and C, would request a virtual circuit between A and B, B and C, and C and A. To the customer, this set of virtual circuits appears like a private network connecting the sites with capacities and performance characteristics that are mutually agreed upon with the service provider in advance. This paper considers the problem of the service provider who, given a set of customer virtual circuit requests each associated with a bandwidth requirement for the forward and reverse direction, must determine what paths through the backbone network the virtual circuits should follow: we refer to this as the primary routing problem for permanent virtual circuits.

This paper evaluates several strategies for primary routing of permanent virtual circuits using an extensive set of simulations. The network is modeled as having a centralized controller which makes all routing decisions. Customers submit requests to set up virtual circuits in sets. Each virtual circuit request consists of a source, a destination, a forward bandwidth and a reverse bandwidth. The duration of the circuit is assumed to be infinite. The service provider must route the virtual circuits or, if there is insufficient capacity, indicate that some of the circuits cannot be routed. We are interested in the relative performance, in terms of the amount of bandwidth routed, of different routing strategies for primary routing. If one routing strategy is able to route substantially more bandwidth on the backbone network than another, the customers ultimately benefit since the costs of expensive facilities are shared with a larger base of users.

Several factors make primary routing for permanent virtual circuits complicated. Routing decisions must be made on-line. In other words, when a set of requests arrive, the routing decisions must be made without knowledge of future requests. The fact that routing decisions must be made on-line can lead to poor performance when a decision to route a virtual circuit over a particular path has, as a result of subsequent requests, caused the network to be used inefficiently. Unfortunately, frequent rerouting cannot be used to correct these inefficiencies since rerouting in virtual circuit networks such as ATM and Frame Relay causes service disruptions to the customer. Finally, a virtual circuit must be routed along a single path in both directions. The requirement can introduce large inefficiencies especially when requests have differing forward and reverse bandwidth requirements.

In addition to primary routing there are two related problems which are relevant to the virtual private network environment. If a link or switching node fails, the service provider may be able to restore service by rerouting the affected virtual circuits: this is the restoration problem. While this paper focuses on primary routing, our results are relavent to the restoration problem. In particular, an efficient primary routing strategy will generally provide greater flexibility to any restoration effort. Furthermore, since our primary routing strategies route *sets* of virtual circuits, they may be used directly for the restoration problem. The second problem related to primary routing is that of admission control. Routing algorithms for virtual private networks are required to pursue a greedy admission control strategy. Specifically, a virtual circuit must be routed as long as sufficient capacity is available. An alternative admission control strategy called *trunk reservation* rejects a virtual circuit if all available routes for that virtual circuit could result in inefficient use of the remaining capacity. In this paper we present a new theoretical result that shows, using competitive analysis, that the use of trunk reservation is fundamental to efficient routing. Previous work analyzing routing under statistical assumptions supports our result [Kru82, Aki84].

Our evaluation considers several different routing strategies. A popular routing strategy simply picks the path with the fewest number of links since that path uses the fewest amount of resources. We call this *min-hop* routing. When multiple paths all have the same number of links one of several tie breaks can be used. The restoration problem suggests that one might prefer the path that maximizes over all links the minimum remaining capacity, so as to allow room for circuits to be rerouted. We call this *max-min* routing. Various algorithms for optimal off-line routing minimize an estimate of the total network delay [BG92]. This suggests choosing the path which minimizes the M/M/1 queue delay. We call this *delay* routing. Finally, the theory of competitive analysis suggests selecting the shortest path in a cost metric where the cost of a link is exponential in the bandwidth currently routed on a link [AAF+93, AAP93]. We call this *exponential* routing. Each of these strategies can be represented as a shortest path or minimum cost routing algorithm. In particular, the strategies differ only in their cost metric.

We evaluate two ways in which the above routing strategies can be adapted to route a set of virtual circuit requests rather than just a single virtual circuit request. The first approach takes a set of requests and processes them sequentially, in decreasing order of bandwidth, in an on-line fashion. We call this the *ordered on-line* approach. The second approach adds a greedy local search in order to improve the route selection. The use of such combinatorial optimization techniques as part of an on-line algorithm is new and may be applicable beyond the area of routing. We call this the *local search* approach.

The simulation results show that exponential and delay routing consistently provide the best performance in terms of the amount of bandwidth routed. Depending on the simulation scenario, exponential and delay routing outperform min-hop routing by 0% to 15% and out perform max-min routing by 6% to 30%.

The paper is organized as follows. Section 2 defines the routing strategies evaluated in this paper. The simulation results for each of the routing strategies are presented in Section 3. Section 4 presents our conclusions and goals for future work. Finally, the proofs for our results on the importance of trunk reservation appear in Appendix A.

2. The Algorithms

This section describes the algorithms we evaluate for primary routing of permanent virtual circuits. Section 2.1 describes each of the cost metrics we consider. The two methods used to enhance the algorithms to deal with a set of virtual circuit requests appear in Section 2.2.

2.1. Cost Metrics

Consider the following notation. For link e let c_e represent the capacity of link e. The bandwidth used by the virtual circuits currently routed over link e normalized by the capacity of the link is denoted by f_e^f in the forward direction and f_e^r in the reverse direction. Finally, for virtual circuit request i, $\Delta f_{i,e}^f$ $(\Delta f_{i,e}^r)$ represents the forward(reverse) bandwidth of request i normalized by the capacity of link e.

Min-hop Routing: A simple cost metric that minimizes the amount of resources used by each virtual circuit assigns a constant state-independent cost to each link. Thus this cost metric chooses the path that consists of the fewest number of links. In this cost metric there will often be many paths with the same cost, therefore a tie break is needed. An *arbitrary* tie-break picks the first minimum cost path that is found whereas a *random* tie-break randomly chooses from all minimum cost paths. Finally, a *state-dependent* tie break use the values of f_e^f, f_e^r, Δf_e^f, and Δf_e^r for all the links on the paths in order to break the tie.

Exponential Routing: Recent results in the theory of competitive analysis suggest the use of a cost metric based on an exponential function. Competitive analysis is an important tool used in the theoretical community to analyze the performance of an algorithm when no assumptions are made about the inputs to the algorithm. In particular, competitive analysis compares the performance of an algorithm on each network topology and request sequence to the optimal algorithm for that network topology and request sequence. Let $A(G, \alpha)$ be the amount of bandwidth routed by algorithm A on topology G and request sequence α. (Note, this analysis can obviously be used for measures of performance other than amount of bandwidth routed.) Let $\text{OPT}(G, \alpha)$ be the amount of bandwidth routed by the optimal algorithm for topology G and request sequence α. The performance of algorithm A is the characterized by

$$\text{MAX}_{G,\alpha} \left\{ \frac{\text{OPT}(G, \alpha)}{A(G, \alpha)} \right\}.$$

which is the *competitive ratio* of algorithm A. Since the competitive ratio is based on a worst case analysis, it is clearly a robust, albeit conservative, measure of performance.

Several recent papers show that a cost metric based on an exponential function leads to routing algorithms that have an optimal competitive ratio. Asnpes et al. [AAF+93] consider primary routing of permanent virtual circuits where the measure of performance for the algorithm is not the amount of bandwidth routed, but the amount of capacity used. Specifically, [AAF+93] shows that an algorithm with a cost metric based on an exponential function leads to a competitive ratio of $O(1/\log n)$ where n is the number of nodes in the network. In other words, such an algorithm would need at most $O(\log n)$ more capacity on each link to route the same amount of bandwidth routed by an optimal algorithm. No better competitive ratio is possible [AAF+93]. The function used for the cost of link e in [AAF+93] is $a^{f_e^f + \Delta f_{i,e}^f} - a^{f_e^f}$ where a is a constant[1]. Unfortunately, the analysis in [AAF+93] provides no information about the

[1] The function does not use any information about reverse bandwidth since [AAF+93] does not consider virtual circuit where the forward and reverse bandwidth differ.

In [AAF+93] the constant a is set to $1 + \epsilon$ where $\epsilon < 1$.

Up to a constant, $a^{x+\Delta x} - a^x$ is just the integral of a^y from x to $x + \Delta x$. Thus a^x can be seen as the marginal cost of the link.

performance of an exponential function based cost metric when $O(\log n)$ additional capacity is not available on each link.

In [AAP93] the performance of the exponential function based cost metric is considered in terms of the total amount of bandwidth routed without the use of additional capacity. However, the model in [AAP93] differs from the permanent virtual circuit model since trunk reservation is permitted. In other words, a virtual circuit can be rejected even if sufficient capacity exists to route the virtual circuit. A simplified version of the proof in [AAP93] can be used to show that a cost metric based on the function $a^{f_e^f+\Delta f_{i,e}^f} - a^{f_e^f}$ leads to a competitive ratio of $O(\log n)$ in terms of the amount of bandwidth routed if trunk reservation is permitted.

Unfortunately the $O(\log n)$ competitive ratio from [AAP93] does not extend to permanent virtual circuit routing where trunk reservation is not permitted. In fact, when trunk reservation is not permitted, there is a lower bound which states that any on-line algorithm has a competitive ratio that is at best $\Omega(n)$ in the amount of bandwidth routed. Appendix A provides the proof of this lower bound. Appendix A also shows that any cost metric will achieve an $O(n)$ competitive ratio. The $\Omega(n)$ lower bound in Appendix A along with $O(\log n)$ competitive ratio from [AAP93] suggest that trunk reservation may be fundamental to efficient usage of network resources. It is interesting to note that, statistical analysis of network routing algorithms also points to the fundamental importance of trunk reservation [Kru82, Aki84].

The particular function we use for the cost of a link is

$$a^{(f_e^f+\Delta f_{i,e}^f)} - a^{f_e^f} + a^{(f_e^r+\Delta f_{i,e}^r)} - a^{f_e^r} + c(\Delta f_{i,e}^f + \Delta f_{i,e}^r). \tag{2.1}$$

The constants a and c are varied in order to determine their best values. Intuitively, a is a measure of how quickly the algorithm retreats from minimum hop routing, while the constant c causes the algorithm to favor minimum hop routing when the network is lightly loaded.

Since the amount of bandwidth required to route all virtual circuit requests in not known in advance, [AAF$^+$93] uses a *scaling* technique. This scaling technique first restricts the algorithm to use at most $1/\log n$ faction of the available capacity. Once no more virtual circuits can be routed with that amount of capacity, the amount of capacity the algorithm may use is doubled. This continues until the algorithm uses all of the available capacity. With the use of scaling, the performance of an exponential function based cost metric should be similar to the performance of the cost metric that seeks to maximize the minimum residual bandwidth (max-min routing).

Max-min Routing: The desire to provide a maximum amount of flexibility in the event that a virtual circuit must be rerouted due to a link failure, suggests selecting the path which maximizes the residual bandwidth for the links in the network. This path selection is achieved with a cost metric that assigns the following cost to path P:

$$-\mathrm{MIN}_{e\in P}\left\{\mathrm{MIN}\left\{1-f_e^f-\Delta f_{i,e}^f, 1-f_e^r-\Delta f_{i,e}^r\right\}\right\}.$$

Delay Routing: Packet network often use routing strategies that route packets so as to minimize the total delay experienced in the network [BG92]. Typically, the delay on each link is approximated by the delay of an $M/M/1$ queue. This suggests a cost metric where the cost of a link is based on the $M/M/1$ queue delay. In the delay based cost metric we set the cost of a link for request i to be

$$\frac{f_e^f+\Delta f_{i,e}^f}{1-f_e^f-\Delta f_{i,e}^f}+\frac{f_e^r+\Delta f_{i,e}^r}{1-f_e^r-\Delta f_{i,e}^r}.$$

2.2. Multiple Requests

This section describes two approaches to primary routing of a *set* of permanent virtual circuit requests. Both approaches can be used in conjunction with any of the cost metrics described in Section 2.1.

Ordered On-line: The ordered on-line approach is based on a simple heuristic. For a given network state and source destination pair a virtual circuit request with a low bandwidth requirement generally has more feasible paths than a virtual circuit request with a high bandwidth requirement. Thus, in order to maximize the chance that every virtual circuit request in a set of requests has a feasible path, the ordered on-line approach gives the requests with high bandwidth requirements the chance to choose a path before the requests with a low bandwidth requirement. In particular, the ordered on-line approach sorts the requests in decreasing order of the maximum of the forward and reverse bandwidth. Once sorted, the virtual circuit requests are routed on-line using one of the methods described in Section 2.1. In other words, after the ith request in the sorted list has been routed, the network state is updated with the bandwidth from ith request and then the $(i+1)$st request is routed based on the new network state.

Unfortunately, this approach does not permit the ith request to take any of the $(i+j)$th requests for $j > 0$ into consideration. The following example illustrates some of the resulting problems.

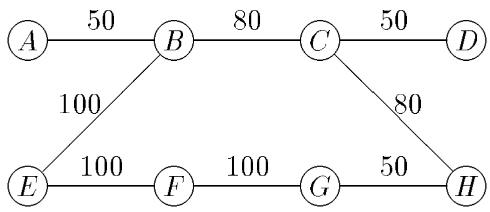


Figure 2.1: Network for Example 2.1.

Example 2.1

Consider the network in Figure 2.1 and the set of requests $(E, H, 46)$, $(F, G, 52)$, $(A, D, 45)$ where the elements of each tuple are the source, the destination, and the bandwidth respectively. The forward and reverse bandwidth is the same for each of the requests. The ordered on-line approach first routes the request $(F, G, 52)$ on the direct path F, G and then routes request $(E, H, 46)$ on the path E, B, C, H. As a result, the request $(A, D, 45)$ has no feasible path. However, it is possible to find a feasible path for all three requests. In particular, if request $(E, H, 46)$ is routed on the path E, F, G, H, request $(A, D, 45)$ could use path A, B, C, D.

Local Search: The local search approach attempts to address the problems outlined in Example 2.1. Rather than considering each virtual circuit request individually, the local search approach considers all requests in a set together. Let p_i be the path used by request i. Then the local search approach used in conjunction with exponential routing attempts to find the *set* of feasible paths that will minimize the following cost function which is the generalization of (2.1) to multiple requests.

$$\sum_e a^{(f_e^f + \sum_{i,e \in p_i} \Delta f_{i,e}^f)} - a^{f_e^f} + \sum_e a^{(f_e^r + \sum_{i,e \in p_i} \Delta f_{i,e}^r)} - a^{f_e^r} + \sum_e \sum_{i,e \in p_i} c(\Delta f_{i,e}^f + \Delta f_{i,e}^r) \quad (2.2)$$

Finding such a set of paths is clearly NP-complete due to the single path constraint for each virtual circuit. We use a greedy local search to find a set of paths whose cost is close to the minimum cost.

The search starts with the set of paths determined using the ordered on-line approach. If the ordered on-line approach cannot find a feasible path for a particular request, it uses the cheapest infeasible path. Denote the set of paths found using the ordered on-line approach by $\mathcal{P}_0$. The set $\mathcal{P}_0$ is said to the *current* set. Let $C(\mathcal{P}_0)$ be the cost of the paths in $\mathcal{P}_0$ based on (2.2). Define $N(\mathcal{P}_0)$ to be the set of path sets that differ from $\mathcal{P}_0$ in at most one path, i.e. at most one virtual circuit has a different path. The local search now finds the lowest cost path set in $N(\mathcal{P}_0)$. Call this path set $\mathcal{P}_1$. If $C(\mathcal{P}_1) < C(\mathcal{P}_0)$ then $\mathcal{P}_1$ becomes the current set. This procedure is repeated to find the lowest cost path set in $N(\mathcal{P}_1)$, etc.. The process stops after the $i+1$st iteration where i is the lowest index such that $C(\mathcal{P}_{i+1}) \geq C(\mathcal{P}_i)$. During the local search the feasibility of the paths is ignored, thus the final set $\mathcal{P}_i$ may have some virtual circuits with infeasible paths. New feasible paths are found for these virtual circuits using the ordered on-line approach. Finally, the virtual circuit requests are routed based on the resulting path set. Simple calculations will show that the local search approach successfully routes all three requests in Example 2.1.

In order to ensure a reasonable running time we limit the number of iteration for the local search to several hundred. The local search is clearly not guaranteed to find the optimal solution since the solution space may have exponential number of local minima and the local procedure terminates after finding the first local minimum. One could explore different regions of the solution space by starting the local search with different (perhaps random) starting points.

3. Simulations

3.1. Simulation Scenario

In order to explore the performance of the different routing strategies, we conducted an extensive set of simulations, varying both the network topology and the characteristics of the request sequences. Where possible, we used data describing existing network topologies and traffic patterns to drive the simulations. However, the simulations explore a number of network topologies and request sequences to minimize the dependence of the results on the specific scenarios studied.

The topology in Figure 3.1 is referred to as the Sparse ATM topology. It describes the initial topology of a planned commercial network for virtual private networks. Figure 3.2 is the topology of an existing commercial network for virtual private networks. We call this topology the Large ATM topology. The Dense ATM topology in Figure 3.1 modifies the Sparse ATM topology by adding additional links between the nodes for which the Large ATM topology has the largest traffic. The capacities of the links are all chosen to be 155 Megabits/sec, which corresponds to existing SONET OC-3 service.

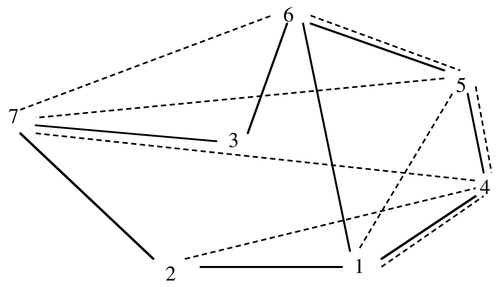


Figure 3.1: The Sparse ATM and Dense ATM topologies. The Sparse ATM topology includes only the solid edges.

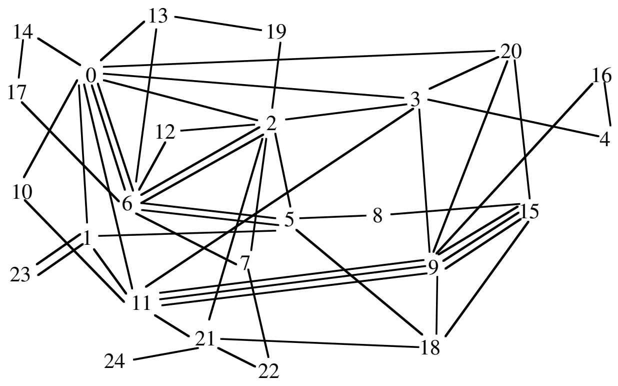


Figure 3.2: The Large ATM topology

Each simulation is driven by a sequence of virtual circuit request sets, called a run, created by a generator. The runs depend to some degree on the network topology, but can also be parameterized independently along the following dimensions:

traffic matrix Each network has a traffic matrix which describes the relative amount of traffic between pairs of nodes in the network. The traffic matrix for each topology is based of data from the Large ATM network. The traffic matrices can be varied by specifying the TM% parameter. If TM% $= x$ then the relative amount of traffic between every pair of nodes is be randomly changed by $\pm x\%$ from its baseline value.

bandwidth The bandwidth of an virtual circuit request is chosen independent of its source and destination. The bandwidth is drawn either from a continuous distribution or from one of six discrete distributions. The discrete distributions are described in Figure 3.3. The continuous distribution is a hyper-exponential distribution. This distribution weighs requests of smaller bandwidth more heavily, however it has a long tail meaning that higher bandwidth requests do occur. The mean of the hyper-exponential distribution, denoted by the parameter MB, is varied between 64 Kbps and 15 Mbps. The coefficient of variation is 30. Initially, the forward and reverse bandwidths are the same.

net traffic flows Since all of its connections have the same forward and reverse bandwidth, the Large ATM network does not have any net traffic flow between any pair of nodes. However, there are many situations such as video servers where one might expect net traffic flows. Hence, there are two parameters that can be used to induce net traffic flows between node pairs. Consider a nodes i and j where $i < j$. The SD% parameter gives the percentage of time when a virtual circuit between node i and node j has node j as the source and node i as the destination. If SD% $= 50\%$, then nodes i and j are equally likely to be the source.

Which node is the source and which the destination is irrelevant when the forward and the reverse bandwidth are the same. Figure 3.3 shows that four of the six discrete distributions include virtual circuit request where the forward and the reverse bandwidth differ. For the continuous distribution, the DF% parameter is used to generate connections with differing forward and reverse bandwidth. If DF% $= x$, then the reverse bandwidth for $x\%$ of the virtual circuits is reduced to 10% of the forward bandwidth. Recall that the continuous distribution initially has the same forward and reverse bandwidth.

request set size The average number of virtual circuit requests in a set of requests is varied using the MS parameter. This parameter represents the mean of the binomial distribution from which the size of each request set is chosen. The binomial distribution has a coefficient of variation of 1.5 and is truncated below at 1.

Each parameter setting is tested with multiple runs using different seeds for the random number generator. The results of these runs are averaged. The number of runs is chosen to

256K/256K	512K/512K	1.5M/1.5M	1.5M/64K	6M/6M	6M/64K	10M/10M	10M/64K
20%	20%	50%	0%	5%	0%	5%	0%
10%	10%	50%	0%	15%	0%	15%	0%
20%	20%	25%	25%	2.5%	2.5%	2.5%	2.5%
20%	20%	5%	45%	.5%	4.5%	.5%	4.5%
10%	10%	25%	25%	7.5%	7.5%	7.5%	7.5%
10%	10%	5%	45%	1.5%	13.5%	1.5%	13.5%

Figure 3.3: Discrete bandwidth distributions. The top row indicates a forward/reverse bandwidth. For example, 1.5M/64K means 1.5 Mbps in the forward direction and 64 Kbps in the reverse direction. Each other row represents a bandwidth distribution by indicating the percent of requests that have a specific forward/reverse bandwidth.

provide a 95% confidence interval that is within 1% of the sample mean.

3.2. Performance Measure

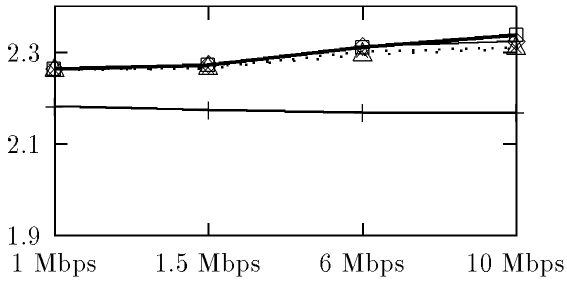
So far, we have focused on the amount of bandwidth routed by an algorithm as its performance measure. Specifically, we are interested in algorithms that maximize the amount of bandwidth routed for all networks and request sequences. However, this performance measure cannot be used directly in our simulations. In particular, the performance measure will lead the simulations to situations where only a small percentage of the requests are actually accepted. This situation arises when most of the network's capacity is currently in use. Clearly, a low acceptance rate is not acceptable in practice. In practice, the network capacity will always be increased to avoid low acceptance rates.

Consider an alternative performance measure which gives the amount of bandwidth routed by an algorithm when for the first time more than 50% of a set of virtual circuit requests is rejected. This performance measure still focuses on maximizing the amount of bandwidth routed, however it avoids the unrealistic, and hence unimportant, scenarios where acceptance rates are low. This performance measure preserves the lower bound of Appendix A. In fact much stronger lower bounds are possible with this performance measure.

3.3. Simulation Results

First we consider the performance of each routing strategy for each topology in what we call the "base case". In this case exponential routing sets the constants a and c to 1000 and 1 respectively. Furthermore we use the ordered on-line approach for dealing with the fact that requests come in sets. In the base case the simulation parameters are set as follows: MB $= 1 Mbps$, TM% $= 0$, SD% $= 50$, DF% $= 0$, and MS $= 10$. The performance of each routing strategy can be observed in Figure 3.4 for the Sparse ATM topology, Figure 3.5 for the Dense ATM topology and Figure 3.6 for the Large ATM topology. In particular consider the bandwidth values on the left hand graphs where the value of the x-axis is $1 Mbps$.

In the Sparse ATM topology, similar performance is achieved by exponential routing, delay routing and min-hop routing. The max-min routing strategy routes approximately 4.5% less bandwidth than exponential routing. In the more complex Dense ATM network max-min routing slips to be 20% worse than exponential routing while min-hop routing with a random tie break is now 3% worse than exponential routing. Finally, in the Large ATM topology min-hop routing with a state dependent tie break[2] and with a random tie break are now respectively 6.5% and 10% worse than exponential routing. Although the detailed results are not presented in the figure, we note that min-hop routing with an arbitrary tie break is up to 15% worse than exponential routing while pure random routing accommodates up to 60% less bandwidth than exponential routing.



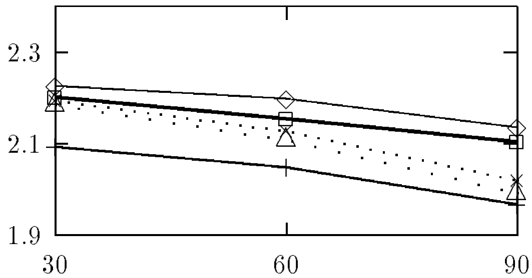


Figure 3.4: Performance in 1000000 ATM cells/sec for the Sparse ATM topology. All parameters are as in base case except that MB is varied in left graph and SD% = 80 and DF% is varied in right graph. Key: exponential: ◇, delay: □, min-hop with state dependent tie break: ×, min-hop with random tie break: △, max-min: +.

Since min-hop routing results in paths that use the fewest amount of resources, the results showing that exponential and delay routing outperform min-hop routing may seem counter

[2] We use the delay function as the state dependent tie break.

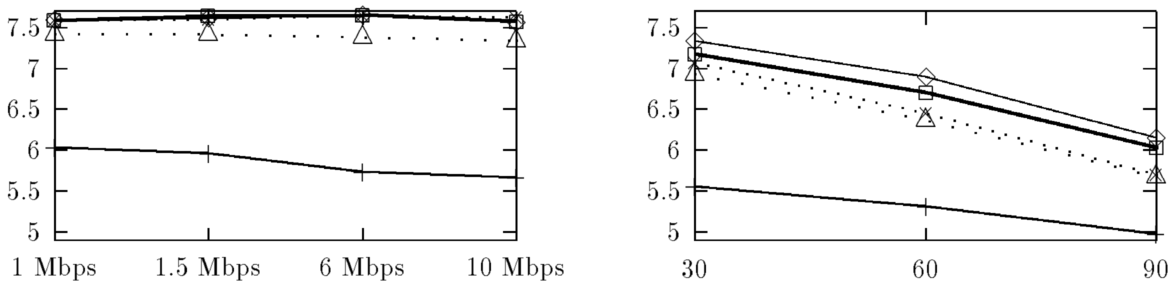


Figure 3.5: Performance in 1000000 ATM cells/sec for the Dense ATM topology. All parameters are as in base case except that MB is varied in left graph and SD% = 80 and DF% is varied in right graph. Key: exponential: $\diamond$, delay: $\square$, min-hop with state dependent tie break: $\times$, min-hop with random tie break: $\triangle$, max-min: $+$.

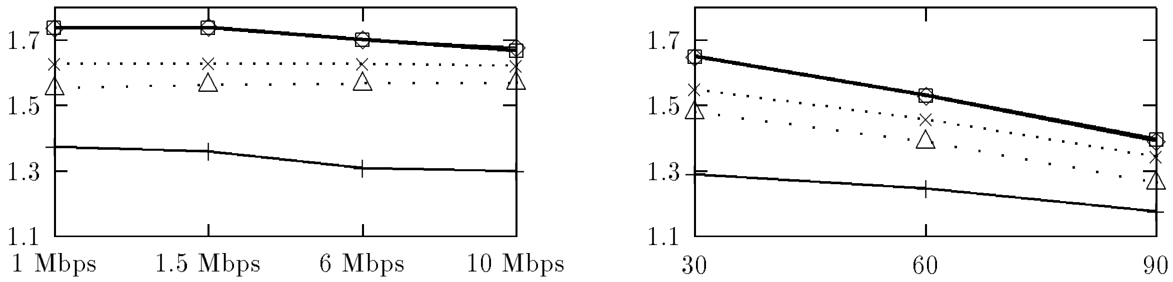


Figure 3.6: Performance in 10000000 ATM cells/sec for the Large ATM topology. All parameters are as in base case except that MB is varied in left graph and SD% = 80 and DF% is varied in right graph. Key: exponential: $\diamond$, delay: $\square$, min-hop with state dependent tie break: $\times$, min-hop with random tie break: $\triangle$, max-min: $+$.

intuitive. The principle behind the delay and the exponential function is that they behave similar to min-hop routing under light loading conditions. However, when links become heavily loaded exponential and delay routing start choosing long lightly loaded paths in favor of shorter heavily loaded paths. The potential advantages of this approach are illustrated in the following example.

Example 3.1

Consider the network in Figure 3.7. Let the capacity of each link be 10. Furthermore, the links from A to B, from B to C and from C to D already carry 9 units of bandwidth in both directions while the remaining links currently carry no bandwidth. Now consider a virtual circuit request with source A, destination D and bandwidth 1 in both directions. Min-hop path routing would pick the path passing through the edge from B to C, thus blocking any

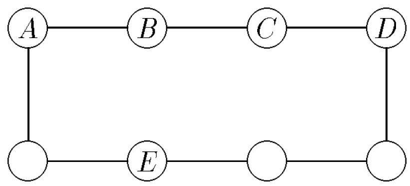


Figure 3.7: Network for Example 3.1.

future request from B to C. In contrast, exponential routing would pick the path through node E thus preserving the ability to accepts requests from B to C.

Returning to the left hand graphs in Figures 3.4, 3.5, and 3.6, observe the the relative performance of the routing strategies is largely unaffected by the increase in the mean bandwidth. The only exception is the decrease in the performance of max-min routing relative to the other routing strategies in the Sparse ATM topology. Increasing the TM% parameter causes the traffic matrix to be less well matched with the network topology. Consequently the performance of the routing strategies should be inversely related to the value of the TM% parameter. The simulations confirm this effect. As with the case of increasing the mean bandwidth of the requests, the relative performance of the routing strategies remains unaffected.

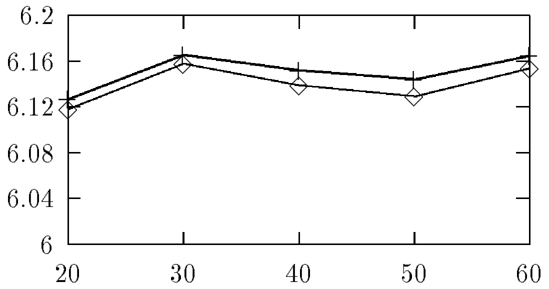


Figure 3.8: Performance in 1000000 ATM cells/sec for the Dense ATM topology. All parameters are as in base case except SD% = 80, DF% = 90 and the MS parameter is varied. Key: ordered on-line: $\diamond$, local search: $+$.

The right hand graphs in Figures 3.4, 3.5 and 3.6 examine the effects of introducing net flows. In the simulations shown in the figures, the SD% parameter is set to 80, while the DF% parameter is varied from 30 to 60 to 90. As the DF% parameter increases larger net flows are introduced. The simulation results for the Sparse and Dense ATM topologies show that, in comparison to the base case, the relative performance of exponential and delay routing increase substantially with respect to min-hop routing with a state-dependent tie break. In particular in the Dense ATM topology with DF% = 90 exponential routing accommodates 7.5% more bandwidth than min-hop routing with a state-dependent tie break. Recall that these two routing strategies had essentially identical performance in the base case. The relative decrease in performance observed for min-hop routing is due to the fact that min-hop routing, unlike exponential and delay routing, does not take the fact that certain links may be carrying much more traffic in one direction than the other into account. The simulations for the Large ATM topology however, do not show a relative increase in the performance of exponential and delay routing with respect to min-hop routing with a state-dependent tie break. This can be explained based on the interaction between the Large ATM topology and the traffic matrix used to generate the source and destination of the virtual circuit requests.

The discussion so far has focused on the continuous bandwidth distribution. Without presenting the detailed results, we note that the use of the discrete distributions produces each of the effects observed for the continuous distribution.

The next set of simulations evaluate the effectiveness of using the greedy local search as a way to deal with the fact that virtual circuit requests arrive in sets. The simulations shown in Figure 3.8 have the parameters set as follows: MB $= 1Mbps$, TM% $= 0$, SD% $= 80$, and DF% $= 90$. The MS parameter is varied from 20 to 60. The greedy local search seems to provide little improvement over the simple ordered on-line strategy. Thus we use the greedy local search only to avoid the pathological cases described in Section 2.2.

In the simulations presented so far, the constants a and c for the exponential routing are set at 1000 and 1 respectively. These values were arrived at using simulations. The results of the simulations suggest that the constant a and c are fairly forgiving in the sense that exponential routing performs well over a wide a and c values. In particular a can take any value from 1000 to 100000 while c can take any value between $a/10$ to 0. This observation hold over all parameter setting that we tested.

4. Discussion

The simulations presented in this paper show that the use of min-hop routing for on-line routing of permanent virtual circuits can lead to inefficiencies despite the fact that it uses the least amount of resources. In many situations it seems that longer less congested paths are preferable to short congested paths. The exponential and delay functions show how congestion and path length should be traded-off. The similarity in performance between exponential and delay functions is not particularly surprising since the functions a^x and $x/1 - x$ are fairly similar.

There are several avenues of further research that we wish to pursue. In particular we wish to use simulations to validate our theoretical observation about the importance of trunk reservation. Furthermore, the routing strategies investigated so far are all centralized. As virtual circuit networks grow distributed routing algorithms will become important.

5. Acknowledgements

We wish to thank Bala Rajagopalan for the data from the Large ATM network. Furthermore, thanks go to Bala Rajagopalan, Vikram Saksena, Bill Aghinii, and Bob Wentworth for invaluable discussions about the system model and the performance measures for primary permanent virtual circuit routing. Finally, discussions with Yossi Azar and Baruch Awerbuch contributed to the lower bound in Appendix A.

References

[AAF+93] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proceedings of the* 23^{rd} *Annual ACM Symposium on Theory of Computing, San Diego, California*, May 1993.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In 34^{th} *Annual Symposium on Foundations of Computer Science, Palo Alto, California*, November 1993.

[Aki84] J. Akinpelu. The overload performance of engineered networks with nonhierarchical and hierarchical routing. *AT&T Bell Laboratories Technical Journal*, 63(7):1261–1281, September 1984.

[BG92] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, NJ, 2nd edition, 1992.

[Jr.86] M. Hall Jr. *Combinatorial Theory*. Wiley-Interscience Series in Discrete Mathematics, 2nd edition, 1986.

[Kru82] R. Krupp. Stabilization of alternate routing networks. In *Proceedings of IEEE Internation Conference on Communications*, page 3I.2.1, 1982.

[MS87] B.A. Murtagh and M.A. Saunders. MINOS 5.1 User's Guide. Technical report, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, Stanford, CA, 1983 revised 1987. Tech report no SOL83-20R.

[PS82] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 07632, 1982.

[RT85] P. Raghavan and C.D. Thompson. Provably good routing in graphs: Regular arrays. In *Proceedings of the* 17^{th} *Annual ACM Symposium on Theory of Computing, Providence, Rhode Island*, May 1985.

A. Appendix

Theorem A.1 (Lower bound)

Let B be an algorithm for on-line routing of permanent virtual circuits with greedy admission control. Assume that B is deterministic or randomized with an adaptive adversary. Then the competitive ratio of B in terms of the amount of bandwidth routed is $\Omega(n)$, where n is the number of nodes in the network.

Proof. Consider the network G in Figure A.1 consisting of $2n+2$ nodes and links of capacity 1. Call the links between nodes v_i and v_{i+1} for $3 \leq i \leq n-1$ the *v-links* and the links between nodes u_i and u_{i+1} for $3 \leq i \leq n-1$ the *u-links*. To prove the theorem we construct a sequence α of virtual circuit requests such that the amount of bandwidth routed by B is $O(1)$ while the amount of bandwidth routed by an optimal off-line algorithm is $\Omega(n)$. The request sequence α is a concatenation of four request sequences α_1, α_2, α_3, and α_4. All of the requests in α have a forward and backward bandwidth of $1/r$ for some $r > 0$.

Let α_1 be a sequence of r virtual circuit requests from S to D. After B routes the requests in α_1, either the *v-links* or the *u-links* are at least 50% utilized. Without loss of generality assume that the *v-links* are at least 50% utilized. Now, α_2 consists of requests from u_2 to u_0 such that after B routes the requests in α_1 followed by the requests in α_2 the link from u_1 to

u_2 and the link from u_1 to u_0 are 100% utilized. The sequence α_3 of consists of $r/2$ requests from S to D. Since the *v-links* are at least 50% utilized after the requests in α_1 and the links from u_1 to u_2 and u_1 to u_0 are 100% utilized as a result of the requests in α_2, the *v-links* are 100% utilized after the requests in α_3. Finally, α_4 consists of $(n-3)r/2$ requests such that there are $r/2$ virtual circuit requests between v_i and v_{i+1} for each $3 \leq i \leq n-1$.

Since *v-links* are 100% utilized as a result of the requests in $\alpha_1.\alpha_2.\alpha_3$, B is forced to reject all of the requests in α_4. Thus, the total bandwidth routed by B after routing the requests in α is at most 3. An off-line algorithm, however, can route a total bandwidth of $n/2$. Consider the following strategy. Route all requests in α_1 using the *u-links*. As a result, all of the requests in α_2 will be rejected, and all of the *v-links* are still free. As a consequence, all the requests in α_3 and α_4 can be routed. The total bandwidth routed by this strategy is 1 from α_1, 1/2 from α_3 and $(n-3)/2$ from α_4. ■

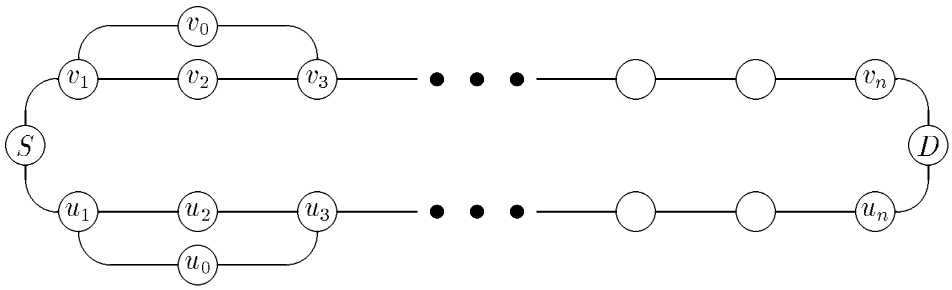


Figure A.1: Network G with capacity 1 on each link.

Define the *scaled bandwidth* of a virtual circuit to be the bandwidth of the circuit times the minimum hop count between the circuit's source and destination. An argument similar to the one used in Theorem A.1 can prove an $\Omega(n)$ lower bound for the competitive ratio of on-line permanent virtual circuit routing algorithms with greedy admission control that seek to maximize the scaled bandwidth routed.

Theorem A.2 (Upper bound)

Let B be any algorithm for on-line routing of permanent virtual circuits with greedy admission control. Denote the bandwidth requirement of the ith request of request sequence α by b_i. Assume that for every request sequence α and network G, there exists a constant $\beta > 1$ such *that the capacity of the lowest capacity edge is at least β times the bandwidth requirement b_i for all i. Then the competitive ratio of B is $O(n)$, where n is the number of nodes in G.*

Proof. Consider any request sequence α and any network with topology $G = (V, E)$. The capacity of link $e \in E$ is given by c_e. Assume that for all i, $\text{MIN}_{e \in E}(c_e)$ is at least βb_i. Let $\mathcal{A}$ be the set of virtual circuit requests in α accepted by algorithm B and let $\mathcal{R}$ be the set of virtual circuits requests in α accepted by the optimal off-line algorithm, but rejected by B. Furthermore, if B accepts the ith virtual circuit request, define P_i to be the path chosen for the ith request by B. Similarly, if the optimal off-line algorithm accepts the ith request, define P_i' to be the path chosen for the ith by the optimal off-line algorithm. Consider the following notation:

$A(i)$: The total capacity accepted by B for all requests up to the ith request: $\sum_{j \leq i, j \in \mathcal{A}} b_j$.

$R(i)$: The total capacity accepted by the off-line algorithm but rejected by B for all requests up to the ith request: $\sum_{j \leq i, j \in \mathcal{R}} b_j$.

$A_e(i)$: The total capacity accepted by B on link e for all requests up to the ith request: $\sum_{j \leq i, j \in \mathcal{A}, e \in P_j} b_j$.

$R_e(i)$: The total capacity accepted by the off-line algorithm on link e for all requests up to the ith request, which B could not have accepted on e due to insufficient capacity: $\sum_{j \leq i, j \in \mathcal{R}, e \in P_j', c_e - A_e(j-1) \leq b_j} b_j$.

Since any path consists of at most n links, $\sum_{e \in E} A_e(i) \leq nA(i)$ for all i.

Assume that the ith request is in $\mathcal{R}$. Since B uses greedy admission control and B rejected the ith request, there exists a link $e \in P_i'$ such that $c_e - A_e(i-1) \leq b_i$. Therefore, $\sum_{e \in E} R_e(i) \geq R(i)$ for all i.

The remainder of the proof assumes that α is finite. Using a simple limit argument the proof can be extended to infinite α. Let k be the last request in α. Consider any link e. Since the optimal off-line algorithm is also bound by capacity constraints, $R_e(k) \leq c_e$. Furthermore, if $R_e(k) > 0$, there exists some request $i \leq k$ such that $c_e - A_e(i-1) \leq b_i$. Since $b_i \leq c_e/\beta$, we conclude that $A_e(i-1) > c_e/\beta$. $A_e(i)$ is by definition monotonically increasing in i, so $A_e(i-1) > c_e/\beta$ implies that $A_e(k) > c_e/\beta$. Using the fact that $R_e(k) \leq c_e$ and $A_e(k) > c_e/\beta$

when $R_e(k) > 0$ and the fact that $A_e(k) \geq 0$, we conclude that $R_e(k) \leq \beta A_e(k)$. Thus,

$$R(k) \leq \sum_{e \in E} R_e(k) \leq \sum_{e \in E} \beta A_e(k) \leq \beta n A(k).$$

The competitive ratio of B is upper bounded by

$$\text{MAX}_{\alpha,G} \left\{ \frac{A(k) + R(k)}{A(k)} \right\}.$$

Using the fact that $R(k) \leq \beta n A(k)$ leads to the desired result. ∎