Routing for Low Call Blocking in High-Speed General Topology Networks

Quan Sun Horst Langendőrfer

Institute of Operating Systems and Computer Networks
Technical University of Braunschweig
Bűltenweg 74/75, 38106 Braunschweig, Germany
Email: {sun,langendf}@ibr.cs.tu-bs.de

Abstract

Maximizing call acceptance probability is one of the most important routing goals in high speed networks. In this paper, we survey some of the existing important routing algorithms aiming at achieving a low call blocking rate for *general topology* networks. A new routing algorithm based on the shortest path algorithm and the competitive routing idea is also proposed in the paper. Extensive simulation experiments were performed to compare the performance of these algorithms. It is shown that the routing algorithms simply based on the routing idea in circuit-switched networks have satisfactory performance only at medium to high network loads. The shortest path algorithm performs well only at low network loads. The EXP algorithm based on the competitive and stochastic analysis can generally lead to low network blocking, but it has the undesirable restrictions of uniform link capacities and uniform call bandwidth requirements. Our proposed algorithm has a desirable low call blocking rate at any network loads.

Key words: routing, general topology network, call blocking rate

1 Introduction

In the past few years, much work has been focused on the routing problem in high-speed networks. One of the most important routing goals ever proposed is the optimization of the global network performance such as call blocking rate [1]. Maximizing the probability of call acceptance is very important to both network providers and users. In this paper, we focus on those routing algorithms aiming at achieving a low call blocking rate for *general topology* high-speed networks.

The routing objective of traditional general topology (data) networks is to minimize quantities such as the average delay [2]. The goal of maximizing the probability of call acceptance in general topology networks is similar to that of traditional circuit-switched complete topology networks [3] [4]. Recently, routing in general topology networks with the mentioned goal has received great attention [5] [6] [7]. The proposed routing algorithms in [5] [6] are an extension of that used in circuit-switched networks with the idea of trunk reservation and controlling alternate paths. The algorithm in [7] is based on the idea of competitive routing recently proposed in [8].

Minimization of call blocking in general topology networks can be achieved by balancing the network load, which is one of the starting points of almost all of the proposed algorithms. Although promoting load balancing can lead to the "packing" problem (also known as "bandwidth fragmentation"), it is not serious for high-speed networks, where the link (residual) capacities are on the order of Gigabits per second [6]. There exist many methods to balance the network load. In circuit-switched networks, we adopt dynamic routing to promote network load balancing. For general topology packet-switched networks, it is a well-known fact that the shortest path routing algorithm with the link length being inversely proportional to the link residual bandwidth can get network load evenly distributed (see e.g. the simulation studies in [9]).

In this paper, we give a new simple routing algorithm based on the shortest path algorithm and the competitive routing idea for general topology networks. The performances of our proposed algorithm and some of the most important existing algorithms are compared through extensive simulation experiments under different network topologies and various traffic patterns. The paper is organized as follows: we first give a short description of the existing algorithms in the next section. Our proposed algorithm is given in section 3. The extensive simulation studies are presented In section 4. Section 5 concludes the paper.

2 The existing routing algorithms

Many new routing techniques for high-speed networks have been proposed in the past few years. The goals of the existing routing algorithms include satisfying Quality-of-Service (QoS) requirements of real-time applications [10] [11] (e.g. delay bound [12], packet loss rate [13]), maximizing the probability of call acceptance (e.g. [7]) and considering some other parameters like constraints of the call admission control functions [9]. In the following, we give a short survey of some of the important existing algorithms focusing on how to achive a low call blocking rate for high-speed general topology networks. Those routing algorithms that are used mainly for complete topology networks (e.g. [14] [15]), or that require large computation time (e.g. [16]) or that need some special implementation techniques (e.g. with the help of a large database containing pre-computed routes [17]) are not considered.

2.1 Controlling Alternate Routing

Sibal and DeSimone [5] have extended the routing idea in circuit-switched networks to general topology packet-switched networks. Their scheme works in the following fashion. When a call request arrives, the primary path from the source node to the destination node is first tried. If sufficient resources on each link of the primary path are available, the call is accepted; otherwise, alternate (secondary) paths are successively attempted. The main contribution of their paper is that a scheme of controlling alternate routing based on state-protection thresholds was proposed. An alternate-routed call can be accepted, only if all links in the alternate path have an available bandwidth greater than the state-protection thresholds. In particular, they showed that if the state-protection level (reservation parameter) is chosen above a certain value we are guranteed to do better than the single-path routing (i.e. calls are permitted to complete only on their primary path) under their Poisson call arrival (at each link) assumptions. In more detail, as long as

$$\frac{B(\Lambda^k, C^k)}{B(\Lambda^k, C^k - r^k)} \leq 1/H \quad \text{for all link } k \qquad (1)$$

accepting an alternate-routed call will necessarily improve on single-path routing, where Λ^k is the primary traffic demand on link k, C^k is the capacity of link k, r^k is the state-protection (reservation) level of link k, B is the standard Erlang blocking function and H is the maximum hop-length over all alternate-routed calls.

2.2 Routing in the plaNET Network

The approach of the plaNET routing algorithm [6] also relies on the use of primary and secondary paths associated with each origin and destination pair (OD pair) in the network. While [5] has not explicitly given the definition of primary and secondary paths, in plaNET routing scheme for a given OD pair, the primary paths are defined to be those with fewest links, all other paths are secondary paths. Links belonging to the primary paths of an OD pair are identified as primary links for that pair, all others are secondary links. Associated with each link are primary- and secondary-load thresholds. A primary path is accepted if the loads on all its (primary) links are below the levels corresponding to primary-load thresholds. A secondary path is accepted if the load levels on all its primary links are below their primary-load thresholds, and the load levels on all its secondary links are below their secondary-load thresholds. Primary- and secondary-load thresholds are typically set to 100% and 95% of the maximum reservable link capacity respectively.

The routing algorithm is a modified shortest path algorithm, where path length is a function of both hop count and individual link lengths. The length of link k is defined as

$$d_k = \frac{C^k}{R^b R^a} \qquad (2)$$

where C^k is the capacity of link k, R^b and R^a are the residual bandwidth of link k before and after adding the new connection respectively.

The length l_p of path P is defined by the pair (n_p, d_p), where n_p is the number of links/hops in P and d_p is the sum of the individual lengths of links in P. Path P_1 is said to be shorter than path P_2, if $n_{p_1} < n_{p_2}$, or $d_{p_1} < d_{p_2}$ for $n_{p_1} = n_{p_2}$.

The routing algorithm functions like the standard Bellman-Ford shortest path algorithm [2]. Computations progress by increasing hop count step by step. At each step (for a given hop count), the algorithm produces paths with minimal total link lengths. The algorithm terminates either when the maximum possible number of hops is reached without finding a feasible path, or at the first hop-count value for which a path with a total length below a specified maximum-length threshold is found [6].

2.3 Shortest Path Routing

The standard shortest-path routing algorithm [2] with the link length being inversely proportional to the link residual bandwidth is expected to be able to balance the network load and therefore can lead to low network blocking. The simulation studies in [9] have shown that this algorithm (in the context of considering the Weigted Fair Queuing (WFQ) service and admission control policy) performs very well regarding the call acceptance probability. The shortest path algorithm has been widely used in general topology packet-switched networks.

2.4 The EXP algorithm

The EXP algorithm developed recently in [7] is based on the notion of competitive analysis (see [8] for a survey). Each link e in the network is assigned a cost that is exponential in its utilization u_e. A call request is accepted, only if there exists a sufficiently cheap path. The algorithm works in the following fashion. For a call request with the source node s, destination node d and bandwidth requirement r, if there exists a minimum-hop path P in the network from s to d, such that

$$\sum_{e \in P} \mu^{u_e} \leq \mu \qquad (3)$$

and every link in P has a residual bandwidth larger than r, then the call request is accepted; otherweise, it is rejected. The μ in (3) is computed as a function of the critical link utilization u^*, i.e. the utilization above which a link should be reserved for single link paths. In terms of u^*, it is easy to obtain

$$\mu = 2^{1/(1-u^*)} \qquad (4)$$

The authors in [7] have developed a novel method to determine u^*: assume that call arrivals are Poisson process and that the call durations are exponentially distributed, then

$$u^* = jr/b \qquad (5)$$

where r is the call bandwidth requirement, b is the link capacity and j is determined by

$$\frac{B(b/r, \lambda^*)}{B(j, \lambda^*)} > 0.5 \qquad (6)$$

where B is the standard Erlang's B formula and λ^* is the arrival rate needed to generate a 2% loss rate on a single link in the absence of any other traffic.

One of the most important design goals of the EXP algorithm is that advance knowledge of the traffic pattern is not required. The restrictions of the algorithm are the same bandwidth requirement of all call requests and the requirement of the same capacities of all links in the network.

3 Our proposed algorithm

From the last section, we see that both the CAR (Controlling Alternate Routing) algorithm and the plaNET (routing) algorithm enjoy the benefits of alternate routing - low average network blocking and good fairness properties at high network loads. Because of its novel admission control scheme, the EXP algorithm can also lead to low network blocking when network load increases, as will be seen in the simulation studies in the next section. However, when network load is low, the shortest path algorithm shows its superiority of balancing the network load. We can expect that the shortest path algorithm will achieve a lower network blocking than the CAR, plaNET and EXP algorithms at low network loads, especially for sparse networks with uneven traffic patterns. The great disadvantage of the shortest path algorithm is the high network blocking at high network loads, because the algorithm has no mechanism of judging the feasibility of the computed paths at certain network load levels.

Our goal is to design an algorithm that enjoys the desirable network load balancing feature of the shortest path algorithm at low network loads and a reasonably low call blocking rate at high network loads. Especially, we developed an admission control scheme to decide if the resulting path computed by the shortest path algorithm should be accepted or not. The proposed scheme is motivated by the concept of competitive analysis [8]. Each link e in the network is assigned a cost that is exponential in its utilization u_e. Let r denote the requested bandwidth and P denote the path computed by the shortest path algorithm. According to [7], if every link in P has a residual bandwidth larger than r and

$$\sum_{e \in P} r\mu^{u_e} \leq \rho \qquad (7)$$

then P should be accepted, where μ and ρ are constants.

In [7] μ and ρ are computed on the undesirable assumptions of the same bandwidth requirements of all calls and the uniform network link capacities. In the following, we give a new mechnisum for choosing μ and ρ without such assumptions. We consider two extrem cases, namely single-link paths and largest-hop count paths. Let R and R' denote the maximum and minimum possible requested bandwidth respectively. It is observed that a single-link path provides the most efficient use of network resources possible, all call requests with a bandwidth requirement not larger than M (M is a constant and $R' \leq M \leq R$) along a single-link path should be accepted. Therefore, we set $\rho = M\mu$. Then, (7) becomes

$$\sum_{e \in P} \mu^{u_e} \leq \frac{M}{r}\mu \qquad (8)$$

Next, consider the largest-hop count paths. As discussed in [9], a routing algorithm should be fair to a large class of requesting calls. Different types of calls (e.g. calls requiring short-hop count paths and calls requiring large-hop count paths) should have roughly equal probability of acceptance. Let $|V|$ denote the number of network nodes, a call requesting a largest-hop count path P should be accepted if every link in P has a utilization less than a given value u^*, then we have

$$\sum_{e \in P} \mu^{u_e} < (|V| - 1)\mu^{u^*} \leq \frac{M}{r}\mu \qquad (9)$$

From (9), we have

$$\mu \geq \left(\frac{r}{M}(|V|-1)\right)^{\frac{1}{1-u^*}} \qquad (10)$$

Because $r \leq R$, we set

$$\mu = \left(\frac{R}{M}(|V|-1)\right)^{\frac{1}{1-u^*}} \qquad (11)$$

If all call requests have the same bandwidth requirement (in this case M=R=r), then (8) is the same as (3) and (11) becomes

$$\mu = (|V|-1)^{\frac{1}{1-u^*}} \qquad (12)$$

The final proposed algorithm is a modified shortest path algorithm with the link length being inversely proportional to the link residual bandwidth. When a call request with the source node s, destination node d and a bandwidth requirement r arrives, our algorithm takes the following steps:

Step 1: those links with the link residual bandwidth of less than r are first pruned (the remaining network graph is denoted by G').

Step 2: if there exists a direct link from s to d in G', then the algorithm terminates and the computed path is the single link (s,d); otherweise go to step 3.

Step 3: compute a shortest path P from s to d in G'. If (8) is satisfied, then P is the resulting path from s to d; otherweise, no path is generated (i.e. the call is rejected).

As seen in step 2, the single-link path is always accepted if enough bandwidth in the link is available (a multicast routing algorithm proposed in [18] with similar idea shows also good performance). In step 3, a computed shortest path is not always accepted, unless the condition (8) is satisfied. In the simulation experiments described in the following section, we set u^*=0.8 to compute μ and it will be found that the proposed algorithm performs very well.

4 Performance studies

We have performed extensive simulation experiments to compare the performance of the routing algorithms described in the last two sections. Due to the following reasons, we have not included the CAR (Controlling Alternate Routing) algorithm in our simulation experiments:

(1) CAR assumes calls of identical statistics. Especially, all calls are assumed to have exponential holding times of *equal* mean length [5]. The calls in most of our experiments had not such restriction.

(2) Both CAR and EXP algorithms work in a similar fashion. They try paths in order of increasing hop count. The difference lies in the admission control criteria. For CAR, a call can be routed along a path, only if all links in this path have an available bandwidth greater than the state-protection threshold. As discussed in [7], EXP has a more rational behavior than CAR in how to accept a call along a path with some highly utilized link(s) and some lightly utilized link(s), because EXP uses a link cost function that is exponential in the link utilization. From this point of view, EXP should perform at least as good as CAR in most network load situations.

In the simulation, we used call rejection probability to evaluate the routing algorithms. The call rejection probability was calculated as the (number of calls rejected)/(number of calls requested) over the last R requests. R was typically set to 500 in our experiments. This call rejection probability was measured as a function of the (normalized) aggregate traffic offered to the network:

$$\frac{1}{h}\sum_{i,j\in V, i\neq j}\lambda_{ij}h_{ij} \qquad (13)$$

where λ_{ij} and h_{ij} are the average number of calls per unit time and average call holding time for calls of the ordered node pair (i, j) (i.e. calls originating at node i and destined for node j) respectively, h is the average call holding time for all calls offered to the network and V is the node set of the network.

All call requests arrived as a Poisson process and the call holding times were exponentially distributed. For simplicity, the holding times of the calls of a given ordered pair were the same, but that of different ordered pairs could be different.

For each simulation experiments, 250,000 call requests (500,000 call requests for the experiments in section 4.3) were offered to the network. For warming up the simulation, the initially 10% of call requests were not considered in computing the call rejection probability. The 95% confidence level was used. 95% confidence intervals for all curves were less than 5%.

In the following, the symbol plaNET stands for the routing algorithm in the plaNET network, EXP for the EXP routing algorithm, SP for the shortest path algorithm and MSP for our proposed algorithm. The link length for SP was also set to be inversely proportional to the link residual bandwidths in the experiments.

4.1 Network with complete topology and uniform link capacities

We first made the simulation experiments for a complete topology network with uniform link capacities. Our simulated network has 8 nodes, there are a pair of unidirectional links transmitting in opposite directions between any two nodes in the network. All links had a capacity of 155 Mbps. All calls had the same bandwidth requirement of 1 Mbps. The average call holding times for calls of different ordered pairs were (randomly) generated from the range [1 minute, 20 minutes]. The offered traffic was uniformly distributed among the network node pairs.

In the simulation experiments, both plaNET and EXP were only allowed to try paths with hop count of 1 or 2. In this situation, the plaNET algorithm worked a bit like the dynamic routing algorithms proposed for circuit-switched networks. The simulation results are given in Fig.1. It is shown that plaNET, EXP and MSP have about the same (relatively low) call blocking rate when network load increases, while the shortest path algorithm performs very bad due to its choosing many paths with large hop counts.

4.2 Sparse network with uniform link capacities

Fig.2 shows the topology of the simulated sparse network. This network was used in [9] [19], and is loosely based on the Internet backbone in the U.S. The network has 28 nodes, 45

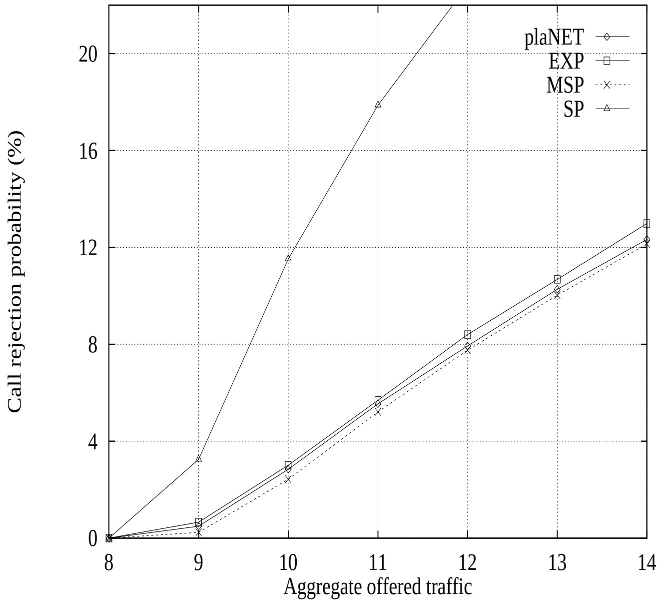

Figure 1: Simulation results for the 8-node network with complete topology

links and a diameter of 7. The average node degree is 3.2. Each link between two nodes in the graph represents a pair of unidirectional links transmitting in opposite directions with capacities of 155 Mbps. All call requests were assumed to have the same bandwidth requirement of 1 Mbps.

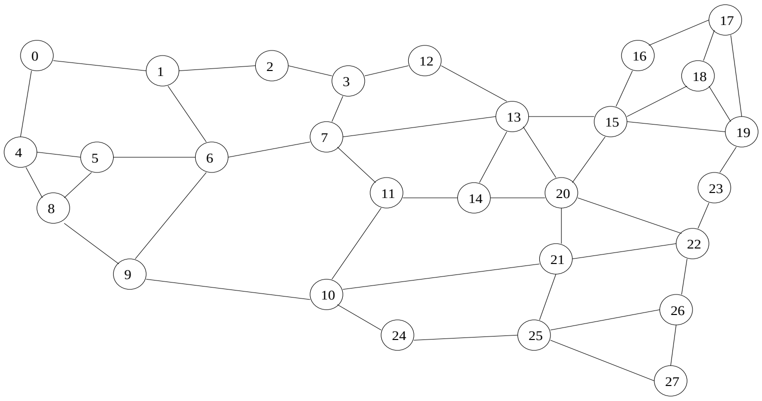

Figure 2: The topology of the simulated sparse network

We used the method described in [20] to compute the k-shortest paths ordered by increasing hop count. There were averagely 11.7 paths between two nodes that could be tried by the EXP algorithm. The same number of paths between two given nodes were also available for the plaNET algorithm. A very large path length threshold was assumed in the implementation of plaNET. The same plaNET algorithm was also used in the experiments of section 4.3.

We first show the simulation results (Fig.3), for which the network topology and the offered traffic are well matched (all offered traffic was for those pairs of nodes that are maximally 2 links apart). The average call holding times for for calls of different ordered pairs were

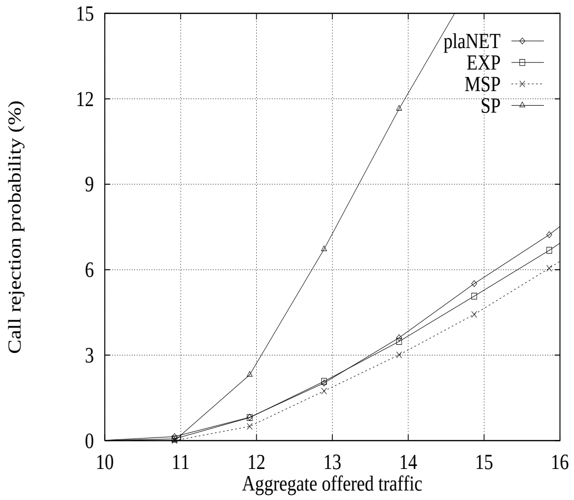

Figure 3: Simulation results for the sparse network: (1) topology and offered traffic are well matched

(randomly) generated from 1 to 15 minutes. It is easy to see that the relative performances of the four routing algorithms have not changed much as compared with that shown in Fig.1. Here, MSP performs slightly better than EXP and plaNET, the shortest path algorithm has the worst performance.

We now see the simulation results (Fig.4), for which the network topology and the offered traffic are not matched (all offered traffic was for those pairs of nodes that are at least 3 links apart). The average call holding times for calls of different ordered pairs were (randomly) generated from the range [1 minute, 15 minutes]. We see that the shortest path algorithm and our proposed algorithm have much better performance than EXP and plaNET. This shows that both SP and MSP have strong capability of balancing the network load.

Finally, we give the performance when the offered traffic is uniformly distributed among the network node pairs. The average call holding times for calls of different ordered pairs were also randomly generated from the range [1 minute, 15 minutes]. Fig.5 and Fig.6 show the results at low and medium to high network loads respectively. It is shown that the shortest path algorithm and our proposed algorithm perform much better than EXP and plaNET at low network load levels. However, when the offered traffic is large, the shortest path algorithm performs very bad, while our proposed algorithm has the best performance. At medium loads, plaNET performs slightly better than EXP, but EXP has a better performance than plaNET at high network loads.

4.3 Sparse network with different link capacities

We now assume the two unidirectional links of the following node pairs (4,5), (5,6), (6,7), (7,11), (11,14), (14,20) and (20,22) in Fig.2 have the capacities of 620 Mbps. The capacities of all other links are still 155 Mbps. In the experiments, heterogeneous traffic was also offered to the network: a randomly generated value from the range (0,5) Mbps was used as the bandwidth requirement of an incoming call request. The average call holding times for calls of different ordered pairs were randomly generated from the range [1 minute, 15 minutes].

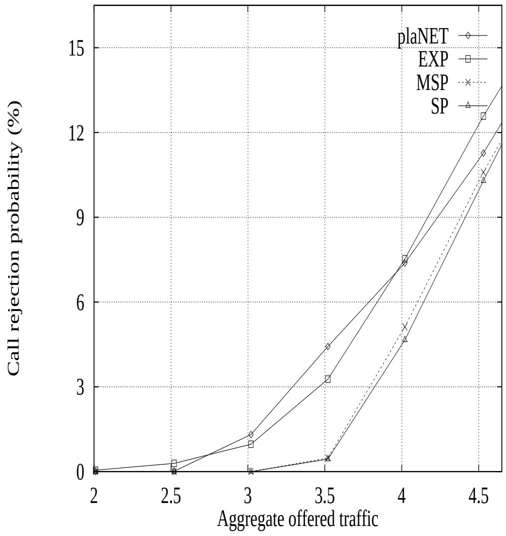

Figure 4: Simulation results for the sparse network: (2) topology and offered traffic are not matched

The offered traffic was uniformly distributed among the network node pairs. Because uniform link capacities and bandwidth requirements are assumed by EXP, we can only compare the performance of plaNET, SP and MSP in this section.

It should be pointed out that in the case of heterogeneous call bandwidth requirements different routing algorithms can not be fairly compared by using the call blocking probability alone. For example, a routing algorithm that generates paths mainly for calls with low bandwidth requirements while rejecting calls with large bandwidth requirements will generally lead to low network blocking but should not be regarded as a good routing algorithm. Thus, we introduce a second measure Thr to compare the different routing algorithms:

$$Thr = \sum_i n_i r_i T_i \qquad (14)$$

where r_i and T_i are the bandwidth requirement and the duration of the call i respectively and $n_i = 1$ if call i is accepted, otherwise $n_i = 0$. The sum of the bandwidth-duration product of (14) can be seen as a measure of the network throughput (see also [8]).

For MSP, we run the experiments for $M = 1$, 2.5 and 4 respectively (see (11)). Fig.7 and Fig.9 show the call rejection probability, Fig.8 and Fig.10 show Thr for 500,000 VC call requests. It is easy to see that the shortest path algorithm and our proposed algorithm performs better than plaNET at low network loads. At some medium network loads, the shortest path algorithm also seems to perform better than plaNET regarding the blocking rate, but it leads to a lower network throughput than plaNET. At high network loads, the shortest path algorithm has the worst performance regarding both blocking rate and network throughput. For any network loads, our proposed algorithm has the best performance. From the simulation results, we also see that the performance of the proposed algorithm is not very sensitive to the choice of M.

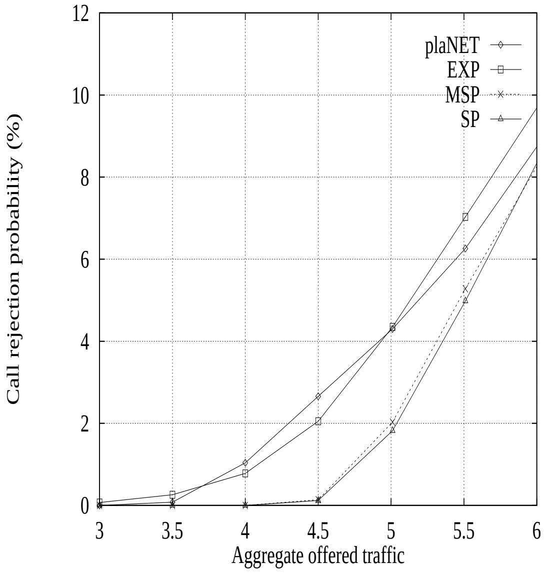

Figure 5: Simulation results for the sparse network at low loads: (3a) offered traffic is uniformly distributed among network node pairs

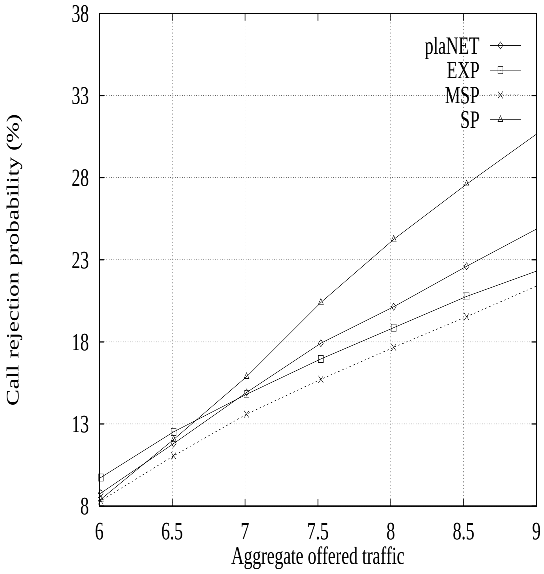

Figure 6: Simulation results for the sparse network at high loads: (3b) offered traffic is uniformly distributed among network node pairs

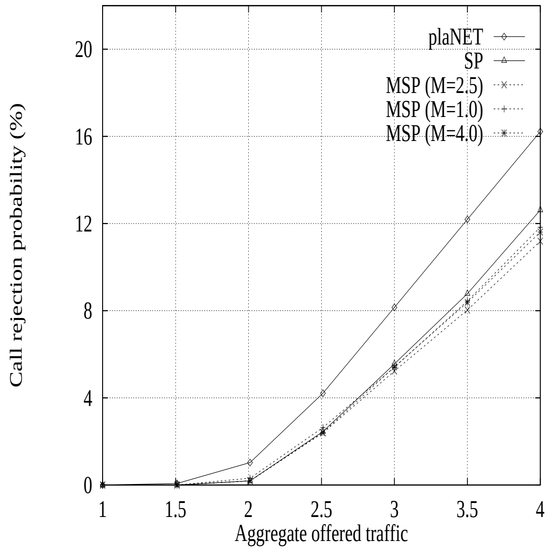

Figure 7: Simulation results for the sparse network at low loads, heterogeneous traffic

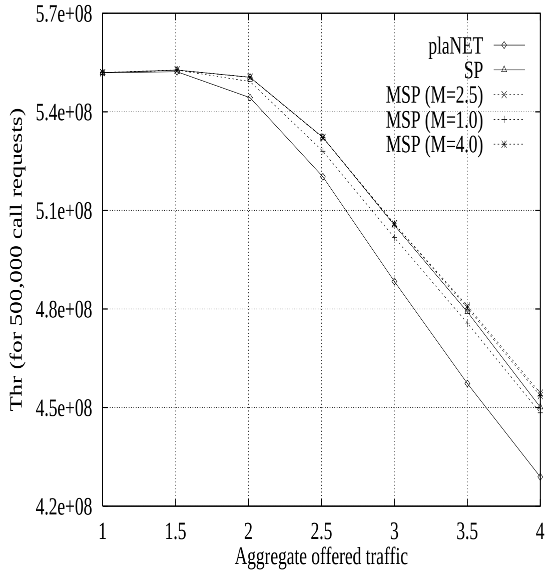

Figure 8: Simulation results for the sparse network at low loads, heterogeneous traffic

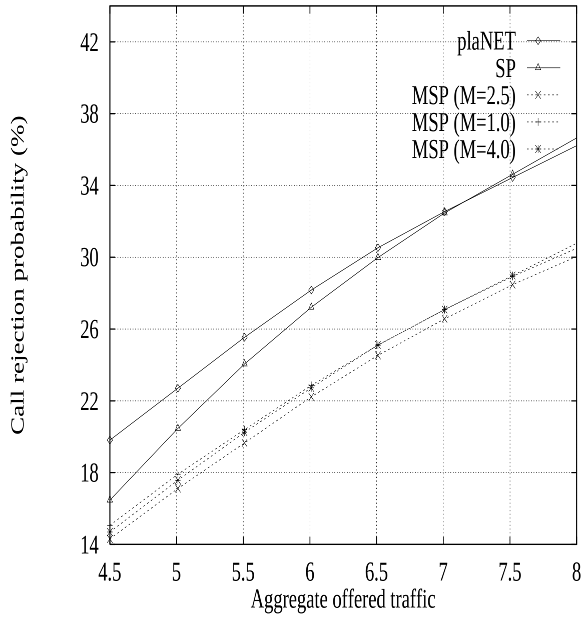

Figure 9: Simulation results for the sparse network at medium to high loads, heterogeneous traffic

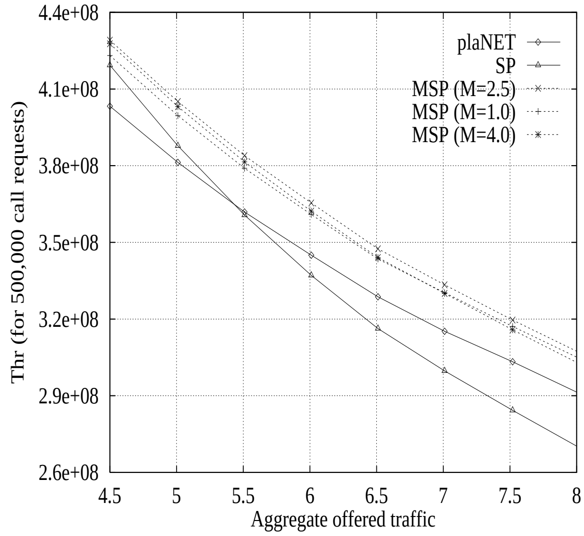

Figure 10: Simulation results for the sparse network at medium to high loads, heterogeneous traffic

5 Conclusions

Maximizing the call acceptance probability is one of the most important routing goals. Most past work was related to circuit-switched complete topology networks. In this paper, we have surveyed several important algorithms aiming at achieving a low network blocking for *general topology* networks. We have performed extensive simulation experiments to compare the performance of these algorithms. A new routing algorithm based on the shortest path algorithm and the competitive routing idea is also proposed in the paper. Our conclusions are:

(1) The simple extension of the routing idea in circuit-switched complete topology networks to general topology networks may not work well. Although the algorithms such as CAR and plaNET have a desirable relatively low call blocking rate at medium to high network loads, their performance at low network loads and for uneven traffic are not satisfactory due to their weak capability of balancing network load for sparse networks.

(2) Because of its novel admission control approach, the EXP algorithm generally performs well. But EXP has the restriction of the same bandwidth requirement of all call requests and the uniform network link capacities.

(3) The widely used shortest path algorithm in packet-switched networks can promote network load balancing efficiently, when the link length is set to be inversely proportional to the link residual bandwidth. The algorithm performs very well at low network loads and for uneven traffic patterns. However, its performance is undesirable at high network loads (and for highly-connected networks) due to its choosing long paths. By introducing the competitive routing idea, our proposed algorithm has a much better performance than that of the shortest path algorithm when network load increases, while it maintains a low call blocking rate at low network loads. In fact, the proposed algorithm performs better than any other algorithms described in this paper.

Finally, it should be pointed out that we have concerned ourself only with the call blocking rate in this paper. Designing of routing algorithms that consider both maximizing call acceptance probability and satisfying the Quality of Service (QoS) constraints for general topology high-speed networks would be the most demanding and challenging future work.

References

[1] Huang, N.F. et al, some routing problems on Broadband ISDN, *Computer Networks and ISDN Systems*, 27, pp.101-116, 1994

[2] Bertsekas, D. and Gallager, R., Data Networks, 2nd ed., Englewood Cliffs: Prentice Hall, 1992

[3] Girard, A., Routing and Dimensioning in Circuit-Switched Networks, Reading: Addison-Wesley, 1990

[4] *IEEE Comm. Mag.*, Special issue on Dynamic Routing in Telecommunication Networks, July 1995

[5] Sibal, S. and DeSimone A., Controlling Alternate Routing in General-Mesh Packet Flow Networks, *ACM SIGCOMM'94*, 24(4), pp.168-179, 1994

[6] Cidon, i. and Guérin, R., Routing in the plaNET Network, *in* Routing in Communications Networks, Steenstrup, M. (Eds.), Englewood: Prentice Hall, 1995

[7] Gawlick, R, et al, Routing and Admission Control in General Topology Networks, Technical Report STAN-CS-TR-95-1548, Stanford University, Stanford, CA, 1995

[8] Plotkin, S., Competitive Routing of Virtual Circuits in ATM Networks, *IEEE Journal of Selected Areas in Communications*, 13(6), pp.1128-1136, 1995

[9] Rampal, S. and Reeves, D.S., Routing and Admission Control Algorithms for Multimedia Traffic, Dept. of Computer Science, North Carolina State University, Raleigh, 1995 (available by anonymous ftp from *ftp.csc.ncsu.edu/pub/rtcomm*)

[10] Lee, W.C., et al, Routing Subject to Quality of Service Constraints in Integrated Communication Networks, *IEEE Network*, pp.46-55, July/August, 1995

[11] Wang, Z. and Crowcroft, J., QoS Routing for Supporting Resource Reservation, Tech. Doc., Dept. of Computer Science, University College London, London WC1 6BT, UK, 1994

[12] Parris, C. J. and Ferrari, D., A Dynamic Connection Management Scheme for Guaranteed Performance Services in Packet-Switching Integrated Services Networks, Tenet Technical Report TR-93-005, Computer Science Division, University of California at Berkeley, 1993

[13] Gupta, S., et al, On Routing in ATM Networks, *in* Routing in Communications Networks, Steenstrup, M. (Eds.), Englewood: Prentice Hall, 1995

[14] Krishnan, K. R. and Ott, T. J., Forward-Looking Routing: A New State-Dependent Routing Scheme, in *Proceedings of the 12th International Teletraffic Congress*, Torino, Italy, 1988

[15] Hwang, R. H., LLR Routing in Homogeneous VP-based ATM Networks, in *Proceedings of INFOCOM'95*, pp.587-593, 1995

[16] Kolarov, A. and Hui, J., Least Cost Routing in Multiple-Service Networks, in *Proceedings of INFOCOM'94*, pp.1482-1489, 1994

[17] Bahk, S. and Zarki, M. E., Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks, *ACM SIGCOMM'92*, 22(4), pp.53-64, 1992

[18] Sun, Q. and Langendőrfer, H., Routing Multipoint Connections in VP-based ATM Networks, in *Proceedings of International Conference on Communication Technology (ICCT'96)*, pp.482-485, 1996

[19] Murakami, k., Near-Optimal Virtual Path Routing for Survivable ATM Networks", in *Proc.* of *INFOCOM'94*, pp.208-215, 1994

[20] Vasilakos, A, et al, On the Use of A Stochastic Estimator Learning Algorithm to the ATM Routing Problem: A Methodology, in *Proceedings of 12th International Conference on Computer Communication*, pp.571-576, 1995