Behaviors and Effectiveness of Rerouting: a Study

Mun Choon Chan
School of Computing
National University of Singapore
chanmc@comp.nus.edu.sg

Yow-Jian Lin
Telcordia Technologies
yjlin@research.telcordia.com

***Abstract*—**

Rerouting has been used in traffic management to perform dynamic load balancing. The aim of rerouting is to reassign path/bandwidth allocations of current traffic trunks in a network in order to minimize the probability of blocking future resource requests. In this work, we are interested in investigating how the effectiveness of rerouting can be affected by the characteristic of the underlying network topology.

In order to evaluate the blocking performance achievable by rerouting, we established baseline measures through two resource allocation algorithms: a shortest distance path algorithm (SDP), that represents the best common practice without rerouting, and a global rerouting algorithm that is based on a provably ϵ-optimal algorithm for multi-commodity flow problem. We proposed two rerouting algorithms based on the basic SDP algorithm that selects for rerouting either from traffic trunks with the same source-destination pairs (local rerouting) or from all traffic trunks (global rerouting).

Our results show that the effectiveness of rerouting is highly related to the average node degree. As the connectivity of a graph increases, rerouting tends to be more effective. However, rerouting does not always perform better when connectivity is increased. Significant performance improvement only occurs within a relatively small range of connectivities when a rerouting algorithm like SDP-LR is able to find alternative paths and SDP cannot.

Furthermore, local reroutng is sufficient to exploit most of the benefits of rerouting and it is not necessary to utilize much more computationally intensive global rerouting algorithms. Finally, we investigate the rerouting frequency vs. blocking trade-off and show that for local rerouting, the best performance can be achieved by a rerouting frequency of only 30%.

I. Introduction

This paper presents our study of rerouting approaches in Multi-Protocol Label Switching (MPLS) [1] capable data networks. With MPLS, it is possible to support Quality-of-Service (QoS) based on a single routing framework over a broad spectrum of network technologies. Studies on traffic engineering in MPLS have focused on two issues: resource allocation and reliability. The issue in resource allocation concerns the optimal use of network resources, such as bandwidth, in order to reduce the rate of blocking resource requests. The issue in reliability is the provisioning of alternate paths for each traffic trunk, subject to link and node fault assumptions. A traffic trunk is an aggregation of traffic flows of the same class which are placed inside a label switched path. Rerouting a traffic trunk to its alternate path occurs only when a link or a node in its primary path has failed.

While rerouting in the presence of failure has been studied extensively, rerouting as a way to improve resource utilization, thus minimize blocking ratio, has not been actively studied in the context of data network. While rerouting techniques have been applied to circuit switching in telecommunication networks, rerouting in this domain tends to focus on a certain class of network topologies where there is a clearly defined set of *primary* paths. Rerouting often involves simply moving traffic trunks routed on *alternate* back to the *primary* paths [2]. For simple topologies, it is proven that rearranging allocated traffic trunks for every traffic arrival and departure minimizes the blocking probabilities [3]. However, it is not clear how such rerouting would work in a more general network.

Rerouting as a resource allocation mechanism has its share of practical concerns. For example, excessive packet drop and/or reordering can occur during rerouting; finding and moving potentially large number of traffic trunks online is also operationally undesirable [4]. Nonetheless, these concerns are manageable. A packet buffering mechanism at source and destination nodes in the network can reduce packet loss and/or reordering during traffic trunk rerouting. An example of how this can be achieved is presented in [5] for ATM networks. Moreover, carefully designed rerouting selection and execution algorithms can help minimize the number of rerouting and traffic distribution during route switching-over.

We believe that with the proper design and planning, rerouting can be a useful resource allocation tool. It is the goal of this paper to explore how effective rerouting can help improve resource utilization and blocking ratio, and whether rerouting remains effective if the frequency of rerouting is reduced.

In this work, we are particularly interested in how the effectiveness of rerouting can be affected by the characteristic of the underlying network topology. As a result, we studied the performance of the various rerouting algorithms using different network topologies.

Four algorithms are evaluated in this work. The first routing algorithm is a variant of the shortest distance path (SDP) algorithm [6]. The next two algorithms are rerouting algorithms based on the SDP algorithm that allow different degree of rerouting complexities. The fourth algorithm is a global rerouting algorithm that serves as a baseline for the performance of any rerouting algorithm. In order to accept a new request, this global rerouting algorithm attempts to reroute any number of traffic trunk for any source-destination pairs such that enough capacity can be found. The algorithm is based on an ϵ-optimal algorithm for multi-commodity flow problem [7].

Our results show that, the effectiveness of rerouting is

highly related to the average node degree. For graphs that are more sparsely connected, rerouting is not effective. As the connectivity of the graphs increase, rerouting tends to be more effective. However, the amount of improvement varies and higher connectivity does not always imply larger improvement. Rerouting is significantly better only in a relatively narrow range of connectivities. In addition, when rerouting is effective, it is sufficient to consider only a limited set of traffic trunks for rerouting and a rerouting frequency of 30% (over all requests) is sufficient to exploit most of the benefits of rerouting.

The paper is organized as follow. Section II briefly describes related work. Section III presents the system model and the rerouting algorithms, followed by experimental results in Section IV. Concluding remarks are in Section V.

II. Related Work

Rerouting is often used in conjunction with restoration to recover from network element failures and has been extensively studied [8], [9]. The key issue is the interdependence between the selection of alternate routes and the amount of reserved spare capacities. In the event of (single) link link or node failure, there should be sufficient resources to reroute all affected flows to their backup routes.

Another common application of rerouting is in the support of mobile communications over wireless networks. In mobile networks, it is often necessary to reroute on-going connections to/from mobile users as these users move among different base stations. The important issues are low hand-off latency, efficient routes and limited disruption to traffic flows [10].

In circuit-switched networks, it is well-known that dynamic routing can provide significant throughput gain over fixed routing. A comprehensive review of dynamic routing can be found in [11]. A way to further improve the throughput of dynamic routing is the use of rerouting. The underlying topology is (usually) a fully-connected mesh network. When a new call p is blocked on its direct path, a call that is using the congested link as its alternate path is randomly chosen and rerouted to its direct path. If rerouting fails, the call p is placed on the least loaded alternate path. If no such path exists, p is blocked.

In this paper, we studied the effectiveness of rerouting on general network topologies. There are no pre-determined path selections, neither are there designations of *primary* and *alternate* paths, as is the case in [13] and [2]. Rather, we rely on the basic principle behind the shortest distance path algorithm for both initial routing on flow arrival and rerouting on departure.

III. System Model and Algorithms

A service provider network consists of a set of edge routers and core routers. Bidirectional links of various capacities (link bandwidth) connect the routers. Customer traffic flows arrive at one edge router (i.e, ingress) and leave at another edge router (i.e., egress) through a traffic trunk. In general, flows may require different quality of services (QoS) treatments.

Let the set of edge routes be E and the set of QoS classes be Q. Each *flow class* $FC_i = (s_i, d_i, q_i)$, where $s_i, d_i \in E, s_i \neq d_i$ and $q_i \in Q$, defines a class of flows that share the same ingress-egress pair (s_i, d_i) and the same service quality q_i. Let N be the number of edge routers in the domain and S be the maximum number of QoS classes for each (s_i, d_i) pair. The maximum number of flow classes in the domain is $N \times (N-1) \times S$.

A bandwidth broker is responsible for assigning a path, thus the bandwidth of each link along the path, to carry the traffic of each flow.

We assume that a set of feasible paths exists for supporting flows of each class FC_i. In a practical implementation, network administrators may prefer to consider only a subset P_i of the paths feasible to the class FC_i.

We consider only the rerouting of traffic trunks, which are placed inside a LSP and assume that changes in traffic trunks occurs at a much slower time scale than customer traffic flows arrival. A traffic trunk request f arrives at an ingress router asking for a path to carrying b units of bandwidth of flow class FC_i traffic. Once the request is granted, f is assigned an LSP $p \in P_i$ provisioned for flow class FC_i. We use $bw(f), fc(f), sp(f)$ to represent the requested/granted bandwidth, the flow class, and the assigned label switched path of traffic trunk f, respectively.

Let $F_i = \{f \mid fc(f) = FC_i\}$ be the set of admitted traffic trunk of the class FC_i.

Let $B(l)$ be the capacity of a link l in the network. At any time the residual capacity $r(l)$ of the link l is $r(l) = B(l) - \sum(bw(f))$, where $f \in \{f_i \mid l \in sp(f_i)\}$. Note that $l \in p$ means the path p includes the link l. To serve a new traffic trunk request f, the bandwidth broker must find a path p provisioned for the flow class $fc(f)$, such that the residual capacity $r(l)$ of every link $l \in p$ is larger than or equal to the requested bandwidth $bw(f)$ of f.

A. Shortest Distance Path (SDP) Algorithm

The path selection algorithm has to balance resource usage during light load and to conserve resources by choosing a short path for each traffic trunk during heavy load. The algorithm chosen for accepting new request arrivals is a utilization-based shortest path algorithm called the *shortest-distance path* algorithm (SDP) [14]. The path cost function is slightly different from that of [14]. Instead of letting the cost of link l be $1/r_l$, where r_l is the residue capacity, we define our link cost to be $e^{(-(r(l)/B(l)))}$. This allows us to incorporate elements of widest-shortest path (WSP) into the SDP algorithm. Let Dist($sp(f_i)$) be the distance of a path $sp(f_i)$.

$$Dist(sp(f_i)) = \sum_{l \in sp(f_i)} e^{-(r(l)/B(l))} \tag{1}$$

SDP finds a feasible path with the shortest (or smallest) path distance for a given traffic trunk request f of class $fc(f)$. If the bandwidth broker can find one such path, it accepts the request and deducts the allocated bandwidth from the residual

capacity of each link along the path. Otherwise, the broker blocks the request.

If rerouting is permissible, a request that is blocked based on current residual bandwidth may be accepted by moving the assigned path of another flow.

In the rest of this section, we present three rerouting schemes. Two of the rerouting schemes are used in addition to SDP to improve performance.

B. SDP with Local Rerouting (SDP-LR)

The SDP-LR algorithm relies on SDP to find a path for new traffic trunk request. In addition, upon each traffic trunk departure, it attempts to reroute a traffic trunk of the *same flow class* to a shorter distance path. Note that the distance reflects the degree of congestion along a path. Consider the case when a traffic trunk $f_d \in F_i$ departs. With the availability of the new bandwidth, we find a traffic trunk for rerouting in the following way.

```
SDP-LR(δ)
for each traffic trunk f in F_i
   calculate the distance Dist(sp(f))
   find the new path, sp'(f), using SDP
    assuming f is rerouted
   calculate the new distance Dist(sp'(f))
select f_max where Dist(sp(f_max)) -
   Dist(sp'(f_max)) is the largest
if Dist(sp(f_max))-Dist(sp'(f_max)) > δ
   reroute f_max
```

This rerouting procedure has a number of advantages. First, it is relatively simple to implement and run. Next, there is a maximum of only one reroute per departure. (The total number of departures is smaller or equal to the total number of arrivals). Recall that rerouting traffic trunks within the same flow class in an MPLS network can be done by simply changing the label on each incoming packet. The disruption to existing traffic is thus minimum and the actual rerouting process is fairly simple and straightforward. By rerouting the path with the largest decrease in distance (or cost), we achieve the goal of reducing the overall network resource usage.

Finally, it is not necessary to perform rerouting on every departure. For example, if the decrease in path distance of f_{max}, δ, is marginal, it may not be worthwhile to perform rerouting. In Section IV, we investigate ways to reduce the number of rerouting by performing rerouting only when δ is larger than a tunable threshold.

C. SDP with Global Rerouting (SDP-GR)

SDP-GR is very similar to SDP-LR. The only difference being the candidate set for rerouting. Recall that the set of candidates for rerouting in SDP-LR is restricted to the set of traffic trunks from the same flow class. Therefore, the utility of SDP-LR may be limited if the average number of traffic trunks in a flow class is small or a better f_{max} can be found from other flow classes. Therefore, in SDP-GR, the candidate set is increased by considering *all existing traffic trunks* instead of just traffic trunks from the same flow class. However, expanding the candidate set to all traffic trunks requires substantially more searching and the rerouting process is also more difficult to handle without flow disruption. Nevertheless, it is instructive to consider the potential performance improvement of SDP-GR.

D. Rerouting Using a Fast Approximation Algorithm for Multi-commodity Flow Problem (MCR)

As a comparison to the proposed rerouting algorithms, we also considered a global rerouting algorithm based on a fast polynomial approximation algorithm for multi-commodity flow problem [7]. In order to use the solution of a multi-commodity flow problem for flow placement, we map each flow class and its aggregated demand into a commodity and its associated demand. Finding an optimal route placement thus becomes equivalent to finding an optimal solution for multi-commodity flow problem. With a large number of flow classes (commodities) and flow requests for each flow class, the fast approximation algorithm allows us to obtain an ϵ-optimal solution in a reasonable time frame. Note that if there is a feasible route placement for all flows, then the optimal solution to the multi-commodity flow problem will be one of them. An ϵ-optimal solution will be a feasible route placement most of the time, too, provided ϵ is small enough.

Our rerouting algorithm is thus as follows. MCR is invoked when we failed to find a path for a new request based on residual bandwidth using SDP. The number of traffic trunks that may have to be rerouted can be large and potentially all existing traffic trunks need to be rerouted. In addition, using MCR, flows within a flow class may be splitted among different paths. Due to space limitations, we will not present the details of the approximation algorithm. Interested readers should refer to [7].

It is important to mention while MCR is an optimal algorithm for placing all active flows known at the time of a new request, it is only a local optimal. Over the lifetime of many requests, MCR may not provide the minimum blocking probability. This is because if MCR allows the admission of a traffic trunk by moving existing trunks to *longer* routes, while other algorithms rejected the request, there can be blocking of more future requests since the network resource is used less efficiently with longer routes.

IV. Simulation Results

The simulation program implements all 4 algorithms presented in Section III and is written in C.

In order to investigate the impact of topology on rerouting, we used two different set of topologies, random topologies and hierarchical topologies. Results for random graphs are presented in Section IV-A and hierarchical graphs in Section IV-B. Since rerouting comes at a cost to the network, in Section IV-C, we studied how rerouting frequency can be reduced and the trade-off between rerouting frequency and blocking ratio.

In the simulations, traffic trunks within a flow class arrive as a Poisson process and have exponential holding time with mean of 1 time unit. All traffic trunk requests are of unit size 1. While this is a simplification, when there are a large number of requests per traffic class, this is a reasonable assumption. The arrival rates are different for different sets of experiments (or graphs) and are chosen so that the blocking ratios are between 0% to 10% in most of the simulations.

All link capacities are of integer units. Simulation duration is 5000 call arrivals. Statistics for the first 2000 calls are ignored and only the last 3000 calls are considered.

A. Rerouting with Random Topologies

This section presents the effect of rerouting under different load conditions using random topologies.

All random graphs have 20 nodes and the three set of random graphs have 40, 80 and 190 bi-directional links. Therefore, the first two sets of graphs have an average degree of 4 and 8. 10 graphs are generated in these two sets. There is only one element in the last set, where all nodes are connected forming a complete graph.

For each of these random graphs, we experimented with the 4 different algorithms presented in Section III.

In each simulation, there are 10 source-destination pairs, all with the same Poisson arrival rate. The holding time is exponentially distributed with mean of 1 time unit. The call arrival rates for the set of random graphs with 40 links is set to 50 calls per time unit and the link capacities are set to 50. The call arrival rates for the set of random graphs with 80 links is set to 80 calls per time unit and the link capacities are set to 30. Finally, for the complete graph, the call arrival rate is 80 calls per time unit and the link capacities are set to 10. In the simulation, all traffic trunks are unidirectional.

For SDP-LR and SDP-GR, rerouting is performed on departure as long as there is at least one traffic trunk whose distance can be decreased by at least 0.5 through rerouting ($\delta = 0.5$). For MCR, $\epsilon = 0.01$. This value should be small enough since the largest link capacity is 50.

Figure 1 shows the blocking ratio with 10 different graphs using 40 links. In 7 of 10 cases, the performance of SDP is almost the same as SDP-LR and SDP-GR. Only in random graphs 2, 4 and 8 are there a significant difference in blocking ratio. This shows that for graphs with limited connectivity, rerouting may not provide substantial benefits. The average improvement of SDP-GR and SDP-LR over SDP is about 3.8% and 2.5% respectively. The performance of MCR is slightly worse than SDP and as explained in Section III-D is due to the inefficient use of network resources by long routes.

Figure 2 shows the blocking ratio with 10 different graphs using 80 links. With a higher degree of connectivity, it can be observed that the performance gap between the rerouting algorithms and SDP have increased. Average blocking ratios over all cases for SDP-LR and SDP-GR is lowered by 5.9% and 6.2% respectively over SDP. MCR also performs better than SDP by 4.6%. In fact, MCR performs the best when blocking ratio is lower, for example in graphs 4, 8 and 9. This

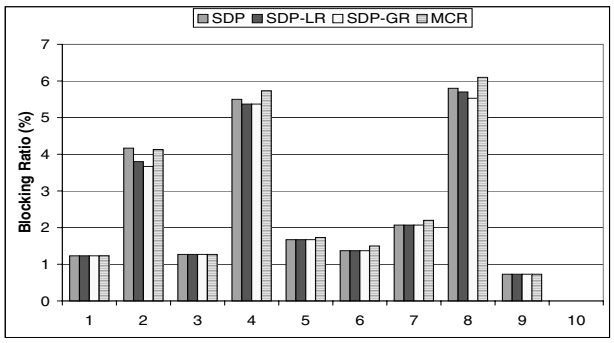

Fig. 1. Blocking ratio of different graphs with 40 links

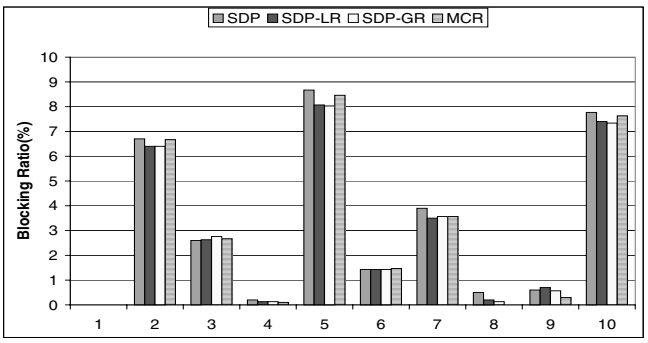

Fig. 2. Blocking ratio of different graphs with 80 links

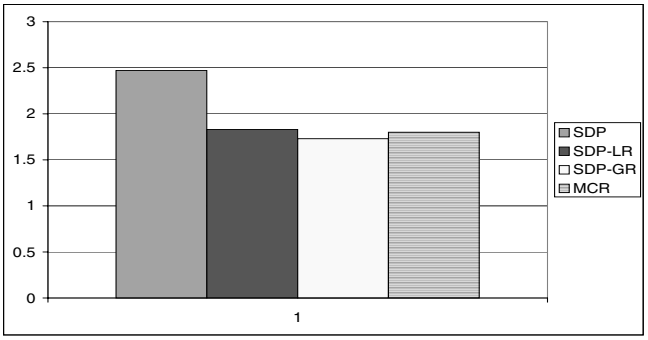

Fig. 3. Blocking ratio of a complete graph

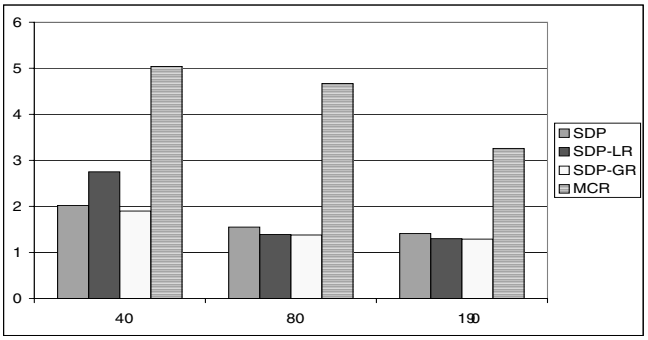

Fig. 4. Average Hop Count

shows that when the network is only slightly congested, MCR is able to better locate alternate routes, and slight inefficiency in resource allocation does not cause more rejection of future requests.

Finally, in Figure 3, we observed the biggest difference between SDP and all there rerouting algorithms using a complete graph. Reduction of blocking ratio from SDP is 26%. This shows that rerouting is most beneficial in highly connected network since it is able to take advantage of the alternative paths without causing potential bottlenecks for other traffic classes.

Taking all together, Figures 1 to 3 clearly show that the effectiveness of rerouting is strongly influenced by the underlying network topologies. A network with high degree of connectivity, for example, the network connecting the core telephone switches, is able to take advantage of rerouting, while a more sparsely connected network may not. In addition, the performance gain of SDP-GR over SDP-LR is not significant.

One way to explain the relationship between topology and rerouting is to look at the average hop count of paths computed. Figure 4 shows the average hop count of routes generated by the different algorithms using different topologies. As the connectivity increases, the average hop count decreases. For example, using SDP, the average hop count decreases 30% going from using 40 links to 190 links. Routes with shorter hop utilize the network more efficiently and usually lead to lower blocking. Notice that the average route hop count using MCR is always the largest and is about 130% to 160% longer than the routes computed by the other three algorithms. Hence, while the use of MCR allows more links in the network to be used, and is optimal for a single request, it also uses the bottleneck links more inefficiently and can lead to more blocking in the future.

In the next section, we compare the performance using larger and more realistic graphs. However, due to the computational complexity of MCR and SDP-GR, only the performance of SDP and SD-LR will be considered.

B. Rerouting with Hierarchical Topologies

We used the *gt-itm* topology generation software (http://www.cc.gatech.edu/projects/gtitm) to generate the hierarchical graphs. A two-level hierarchy is used. There are 10 top level hierarchy, each with 10 nodes, giving a total of 100 nodes. The node connectivity within a hierarchy is random. The edge probability at the top level is varied from 0.1 to 1.0 in steps of 0.1, and 10 graphs are generated for each edge probability. An edge probability of 1 means that there is an edge between all node pairs in the hierarchy. The edge probability at the second level is kept constant at 0.6. Note that in order for rerouting to be useful, multiple paths between different source and destination pairs must exist. Hence, the transit-stub model is not used in the rerouting simulations.

The link capacity of all links are fixed at 30 units. There are 10 randomly placed source and destination pairs in the

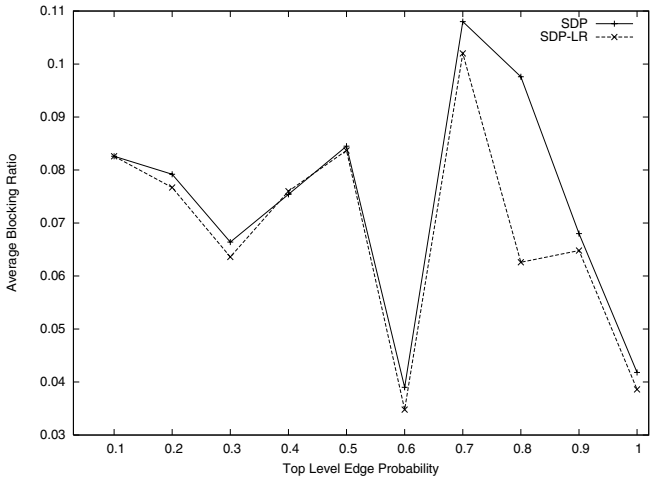

Fig. 5. Average Blocking Ratio

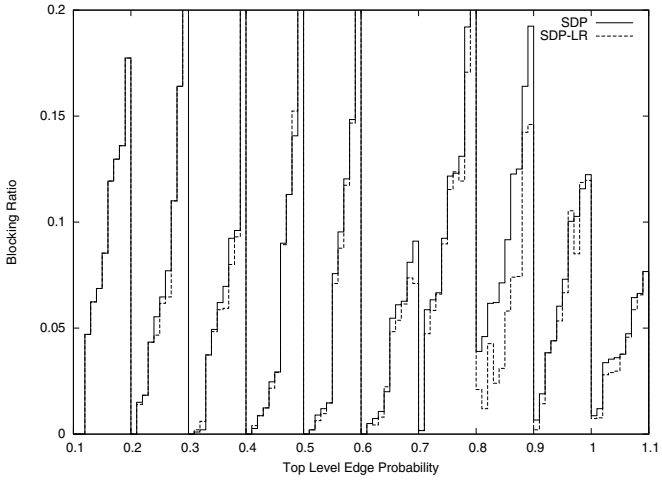

Fig. 6. Blocking Ratios of SDP and SDP-LR

simulation and the traffic loads are selected such that the average blocking ratios for all edge probabilities are between 3% to 10%.

Figure 5 compares the average blocking ratio between SDP and SDP-LR for different top level edge probabilities. 10 different graphs are used for each edge probability. Note that the average values cannot be compared among different edge probabilities since the loadings are different.

For a relatively low edge probability of 0.5 or below, the improvement of SDP-LR over SDP is minor and ranges from -0.8% to 4.2%. When the edge probability is set to 0.1, there is only one possible route across the top level hierarchy. SDP and SDP-LR have identical performance. When the edge probabilities is increased to 0.2 and 0.3, SDP-LR performs slightly better but the improvement is only up to 4.2%. However, as the edge connectivity increases to 0.4 and 0.5, the performance gap goes away, showing that SDP is able to exploit the higher connectivity as well as SDP-LR. For higher edge probability from 0.6 to 1.0, the improvement of SDP-LR over SDP is more significant, and ranges from 4.7%

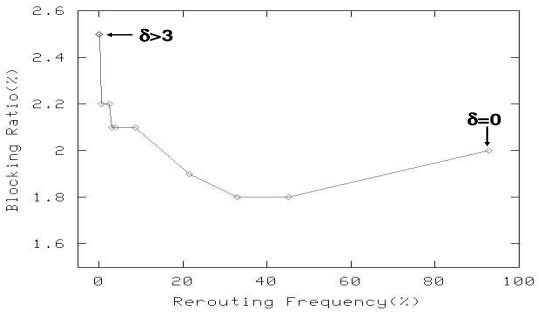

Fig. 7. Blocking Ratio vs. Rerouting Frequency

to 36%. Interesting, the performance gap exhibits a similar pattern. The improvement is largest at 0.8 and much smaller at edge probabilities of 0.6 and 1.0.

Figure 6 compares the actual blocking ratio for each of the 100 simulation runs. The histograms between n_1 and n_2 are for edge probability n_1. For each edge probability, the data is sorted according to the blocking ratio for SDP. It can be easily observed that while there are improvement of SDP-LR over SDP in many cases, the biggest improvement comes when edge probability is 0.8. In addition, in over 100 different runs, SDP-LR performs worst than SDP in only 8 cases.

C. Performance of Rerouting with Different Rerouting Cost

This section reports the trade-off between the number of rerouting attempts and the resulting performance in terms of blocking ratio. SDP and SDP-GR are two extreme cases where in the former case, rerouting never occurs and in the latter case, rerouting almost always occurs on departure. The question is whether it is necessary to perform rerouting on all departures. Clearly, if the traffic trunk targeted for rerouting only uses a path marginally more expensive than the path of departing traffic trunk, rerouting may be unnecessary.

Recall that rerouting is executed only when the distance/cost difference between the original route and the new route is larger than δ. In this section, we simulate SDP-LR and vary δ, the threshold for invoking rerouting at departure, between 0 and 19. Setting δ = 0.0 allows rerouting whenever a path with smaller distance can be found at each departure. On the other hand, setting δ = 19 is equivalent to running SDP since there are 20 nodes in the network and $e^{-(r(l)/B(l))} \leq 1$.

Figure 7 shows how blocking ratio varies with rerouting frequency for SDP-LR in the case of a complete graph (20 nodes, 190 links). When δ is set to 0, up to 93% of the traffic trunks are rerouted on departure. Interestingly, when δ is increased to 0.2 (45% are rerouted), the blocking ratio drops from 2% to 1.8%. This shows that extreme frequent rerouting may actually be bad since the new routes can be highly inefficient in terms of resource usage. When δ is increased beyond 0.5 (33% are rerouted), blocking ratio increases gradually. When δ is set to 3 or larger, the blocking ratio is the same as SDP and no rerouting is performed.

V. Conclusion

In the paper, we study how the effectiveness of rerouting in an MPLS capable data network is affected by the network topology.

The results suggested that, when the average node degree is small, most common practices for route placements such as the shortest distance path (SDP) algorithm yield good performance in terms of blocking ratio and that rerouting may not help much. This is due to the fact that the alternative paths available to each flow class are limited. As a result, rerouting cannot improve much from what algorithms such as SDP have accomplished.

The results also indicate that, when the average node degree increases, the number of available paths increases and rerouting tends to improve the performance compared to the SDP algorithm with no rerouting. However, rerouting does not always perform better with higher connectivity. The largest performance improvement provided by rerouting occurs only within a relatively small range of connectivities when SDP-LR is able to find alternative paths and SDP cannot. In cases when rerouting is effective, local rerouting is sufficient.

Finally, rerouting frequency can be decreased by increasing the reroute threshold δ. While this causes some degradation in performance, changing the threshold δ provides a means of trading off between rerouting frequency and blocking ratio.

References

[1] D. Avduche, J. Malcolm, and J. Agogbua etc., "Requirements for traffic engineering over mpls," RFC 2702, IETF, September 1999.

[2] Eric W. M. Wong, Andy K. M. Chan, and Tak-Shing P. Yum, "A taxonomy of rerouting in circuit-switching networks," in *IEEE Communications Magazine*. IEEE, March 1999, pp. 116–122.

[3] Murat Alanyali and Bruce Hajek, "On simple algorithms for dynamic load balancing," in *Proceedings of the IEEE INFOCOM*. IEEE, 1995.

[4] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter, "Rates: A server for mpls traffic engineering," 2000.

[5] R. Cohen, "Smooth intentional rerouting and its applications in atm networks," in *Proceedings of the IEEE INFOCOM*, 1994, pp. 1490–1497.

[6] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," 1997.

[7] Tom Leighton, Fillia Makedon, Serge Plotkin, Clifford Stein, Éva Tardos, and Spyros Tragoudas, "Fast approximation algorithms for multicommodity flow problems," *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 228–243, April 1995, Preliminary version appeared in STOC '91.

[8] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive surviviable routing," in *Proceedings of the IEEE INFOCOM*, Anchorage, Alsaka, April 2001, IEEE.

[9] M. Kodialam and T. V. Lakshman, "Restorable dynamic quality of service routing," *Communications Magazine*, vol. 40, no. 6, June 2002.

[10] R. Ramjee, T.F. LaPorta, J. Kurose, and D. Towsley, "Performance evaluation of connection rerouting schemes for (atm)-based wireless networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 249–261, 1998.

[11] G.R. Ash, *Dynamic Routing in Telecommunications Network*, McGraw Hill, 1997.

[12] Martha E. Streenstrup ed., *Routing in Communications Network*, Prentice Hall, 1995.

[13] C. Casetti ann R. Lo Cigno and M. Mellia, "Load-balancing solutions for static routing schemes in (atm) networks," *Computer Networks*, vol. 34, no. 1, pp. 169–180, 2000.

[14] Qingming Ma, Peter Steenkiste, and Hui Zhang, "Routing high-bandwidth traffic in max-min fair share networks," in *SIGCOMM*, 1996, pp. 206–217.