Aggregate Based Resource Allocation With Rerouting

Coskun Cetinkaya
Department of ECE
Wichita State University
coskun.cetinkaya@wichita.edu
1845 Fairmount, Wichita, KS 67260-0044

Mun Choon Chan Yow-Jian Lin
Networking Research Laboratory
Bell Laboratories, Lucent Technologies
{*munchoon,yjlin*}*@research.bell-labs.com*

Abstract

This paper studies the effect of rerouting for augmenting aggregate based resource allocation in the trade-off between overhead and utilization. Aggregation is a common approach to address the scalability issue in resource allocation. However, resources committed in bulk may be under utilized while other resource requests are being turned down for lack of resources in some shared links. The aim of rerouting is to free up committed resources for better utilization by reusing resources vacated by terminated flows and by moving existing flows to alternative paths.

Our results show that rerouting improves performance over a wide range of network loads on two different network topologies. In particular, we show that depending on the network load and topology, it is possible to reduce both blocking rate and routing cost.

1 Introduction

Future integrated service networks will support multiple and diverse application with various Quality of Service (QoS) requirements. The main challenge is to provide resources in order to meet the requirements of each connection, and more importantly to meet that goal in an efficient manner. Towards this end, a number of QoS routing algorithms have been proposed to ensure that users' QoS objectives can be satisfied [4]. Such algorithms achieve this goal by calculating a feasible path for each connection. When the number of service requests and frequency of service changes increase with the size of the network and the number of users, signaling cost and state maintenance in the network can eventually become the bottleneck.

The per-flow guarantees can be achieved without per-flow QoS routing by using aggregate resource reservation which can reduce both the signaling load and the amount of state information maintained in the network. We propose a two-tier resource allocation framework which is made up of a set of Edge Bandwidth Brokers (EBB) and a Central Bandwidth Broker (CBB) as a solution. In particular, EBB can maintain a long-time-scale aggregate reservation between a pair of ingress-egress routers. With this existing reservation, individual flows need only signal the EBB which locally accounts for resources along the paths and independently accepts or rejects new flows. Occasionally, when the aggregate reservation is determined to be too large or too small as compared to the actual demand, it can be readjusted via a "bulk" reservation adjustment by signalling the CBB. Thus, the CBB is infrequently signaled to achieve scalability, yet without sacrificing the service model of per-flow guarantees and ideally, with minimal sacrifice in network utilization.

However, resources committed in bulk may be under utilized while other resource requests are being turned down because of lack of resources in some shared links. As a result of this, we encounter lower resource utilization and higher blocking rate. In order to reduce blocking rate and increase resource utilization, we integrate aggregate resource reservation with rerouting which frees up committed resources by moving existing flows to alternative paths and by reusing resources vacated by terminated flows. Rerouting can be done at the flow level as well as aggregate level. We introduce flow level local rerouting and aggregate level global rerouting, present abstract models of both local and global rerouting and evaluate specific implementations of these algorithms. In this paper, we present a new approach for designing scalable provisioning systems with rerouting to achieve better utilization and lower blocking rate. We focus on the fundamental properties of the proposed framework. However, the performance of our approach depends on a number of factors, the most important of one is the traffic characteristic of the underlying flows. For example, if a class' aggregate demand is relatively constant over time, we can achieve high resource utilization and low blocking with small signalling and computation. On the other hand, if a class' aggregate demand oscillates quickly, we would have to choose either rapidly readjusting the reservation to track the demand (we need to signal frequently and lose the scalability), or leaving a gap between the demand and the reservation (we encounter low utilization and high blocking

Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)
1530-1346/03 $17.00 © 2003 IEEE

rate). In the second choice, we can lower the blocking rate with rerouting techniques if the flow's lifetime is relatively longer than execution of rerouting. Our results show that by combining aggregation and rerouting, a resource allocation system with low blocking rate and low signaling cost can be achieved.

With the emergence of Multi-protocol Label Switching (MPLS), use of a single routing framework over a broad spectrum of network technologies, including networks using digital cross connect, wavelength division multiplexing, ATM, Frame Relay and IP, is becoming a viable option. We believe that our work on rerouting has broad applications.

The paper is organized as follows. Section 2 presents related work. Section 3 describes the network architecture and presents the two-tier resource allocation framework and the rerouting algorithms. Section 4 presents the experiment results and section 5 the conclusion.

2 Related Work

In ATM networks, the trade-off between the use of Virtual Path(VP) and Virtual Channel(VC) is well known. The use of VP incurs less signaling load, but has lower performance (higher blocking) due to end-to-end aggregate resource reservation.

In [7], a dynamic Virtual Path bandwidth allocation scheme and the trade-off between transmission efficiency and signaling load processing for multiple VPs sharing a single bottleneck link is studied. In this paper, the size of VP bandwidth is varied in fixed size chunk, and bandwidth changes are triggered by VP utilization thresholds.

In [8], the threshold based VP allocation scheme in [7] is enhanced to allow variable bandwidth changes per allocation. Using a time segmentation technique, the authors were able to show a fairly efficient scheme for computing the bandwidth allocation for each VP bandwidth change such that bandwidth utilization is high and signaling load is low. However, due to the computational complexity of the algorithm, it is not clear how the algorithm can be applied to a real network.

[1] studied specifically the trade-off between the overall network throughput and the processing load on the signaling system. The algorithm produces a set of VPs that handle most of the Virtual Circuit requests until there is insufficient bandwidth in the VP. At that point, hop-by-hop VC signaling is used. The goal of the algorithm is to provide a VP capacity allocation that maximizes a revenue function while meeting the call blocking requirement and signaling constraints.

In IP networks, the scalability problem of Integrated Service is well publicized and many proposals to deal with this problem have been made. For example, PASTE [6] (A Provider Architecture for Differentiated Services and Traffic Engineering) uses MPLS and RSVP as mechanisms to establish differentiated service connections across Internet Service Providers(ISPs). It provides scalability by aggregating differentiated flows into traffic class specific MPLS tunnels and provides the capability for traffic engineering by directing aggregating flows along specific LSP paths.

In the world of circuit-switched networks, it is well-known that dynamic routing can provide significant throughout gain over fixed routing. A comprehensive review of dynamic routing can be found in [2]. A way to further improve the throughput of dynamic routing is the use of rerouting. A common implementation of a rerouting scheme is as follows. The underlying topology is (usually) a fully-connected mesh network. When a call is blocked on its direct path, a call that is using the congested link as an alternative route is chosen randomly, and then it is attempted to reroute to its direct path. If this fails, the new call is routed to the least loaded alternate path. If no such path exists, the call is blocked. For a taxonomy of rerouting in circuit-switched networks, see [10]

Our work is different from the related work described above in the following ways. We combined the element of aggregated resource reservation technique with the rerouting technique in circuit-switched network. We developed a two-tier resource allocation framework with two level rerouting algorithms. The rerouting is performed on aggregate basis in addition to per flow basis in order to reduce signaling load and increase the network utilization. Finally, the algorithm is designed to work with any topology.

3 System Model and Algorithms

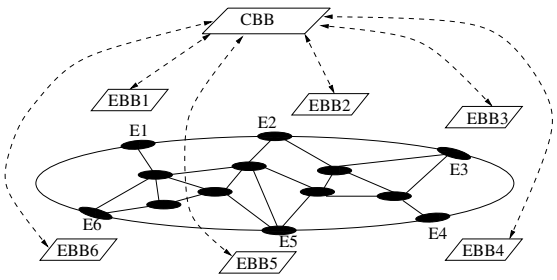


Figure 1. Architecture of two-tier bandwidth broker

This section describes the framework for our rerouting study. We first present a two-tier bandwidth allocation model, and outline the interactions at each tier. We then review the functionality of each tier in detail, with an emphasis on the algorithms for rerouting.

A service provider network of interest consists of a set of edge routers and core routers. Customer traffic flows arrive at one edge router (i.e, ingress) and leave at another edge router (i.e., egress) through a provisioned label switching path (LSP). Flows may require different quality of service (QoS) treatments.

Let the set of edge routers be E and the set of QoS classes be Q. Each *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. 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 that is feasible to the class FC_i. We further assume that the paths in P_i have been pre-provisioned during rerouting. Hence, rerouting of a flow from one LSP to another requires simply changing the label on each incoming packet. For a description of how rerouting can be performed *smoothly* in an ATM network see [3].

The admission control of resource requests is based on a two-tier bandwidth brokers architecture, as depicted in Figure 1. A *Central Bandwidth Broker* (CBB) provisions LSPs and allocates bandwidth along theses LSPs in bulk to aggregated flows of the same flow class, whereas each *Edge Bandwidth Broker* (EBB) assigns individual flows to the LSPs allocated to it by the CBB.

We identify a flow j request f_{ij} as being a request for a path of bandwidth f_{ij} to carry the flow of the class FC_i. Once admitted, f_{ij} is assigned an LSP $p \in P_i$. Let $F_i = \{f_{ij}\}$ be the set of admitted flows of the class FC_i.

Let K_{ip}, $p \in P_i$ be the resource allocated to the flow class FC_i along the path p, and r_{ip} be its residual bandwidth of K_{ip}. The EBB$_s$ at an edge router s manages the set $K_i = \{K_{ip}\}$, for each flow class FC_i such that its ingress router $s_i = s$.

The basic allocation control flow is as follows. When a new flow j request f_{ij} arrives at the ingress router s_i, EBB$_{s_i}$ chooses p such that $r_{ip} \geq f_{ij}$ and admits the request. If none of the LSPs that EBB$_{s_i}$ manages on be half of class FC_i has sufficient residual bandwidth, EBB$_{s_i}$ signals to the CBB, asking for additional bandwidth, b_i, for carrying flows of class FC_i. The bulk size, b_i, is a system parameter for the study. The CBB checks the residual capacity available among the set of paths in P_i to find a path p such that the resource b_i can be allocated. If it succeeds, the CBB will reply to EBB$_{s_i}$ indicating that b_{ip} resource is allocated on path p where b_{ip} means that b_i amount of resources are allocated on path p.

On flow departure, EBB$_{s_i}$ releases resource b_{ip} back to the CBB if residual capacity r_{ip} becomes bigger than b_i by either flow termination or flow rerouting (we will discuss shortly). Note that the CBB could grant multiple increments of bandwidth along the same path p to EBB$_{s_i}$, which in turn accumulates them together in K_{ip}. The EBB could release the bandwidth in same or different increments. In our study we assume that EBB$_{s_i}$ releases them in the same increment.

Just as there are two tiers of admission control, rerouting also takes place at two levels: *local rerouting* and *global rerouting*. Local rerouting occurs within each edge bandwidth broker and is executed by EBB$_{s_i}$. Local rerouting is in the form of re-assigning some existing flows to other paths in order to create enough residual capacity r_{ip} for the new request. It might also free enough bandwidth to be able to release it back to the CBB. Global rerouting takes place at the CBB. Upon receiving a bulk request b_i, the CBB can not satisfy the request if all routes for flow class FC_i are congested. However, it is likely that resources allocated to other flow classes are also utilizing some of the congested links. By rerouting some of the bulks of other classes away from congested links, the CBB can free up resources for b_i request.

The aim of rerouting is to reduce blocking probability and to increase resource utilization in a way similar to reclaiming memory space through garbage collection. Regardless of local or global rerouting, a general rerouting guideline is to reroute a flow (or a bulk) to a new LSP with less or equal resource utilization cost (subject to the cost definition, such as hop count, round trip delay, loss probability, etc.). The goal of moving them to a path with a lesser cost should be obvious. On the other hand, rerouting a flow (bulk) to another LSP with an equal cost is to free up resources at the current path hopefully for others to take advantage of. To reach an optimal global solution it might be necessary to reroute some of them to a higher cost path sometimes. However, it only makes sense to find an optimal global solution when all the flows (bulks), including future requests, are known. Because of the dynamic nature of the on-line traffic and not knowing future traffic, we do not consider getting an optimal global solution.

In the subsections, we will describe EBB, CBB, and the algorithms they run to strike a balance between low signaling cost and low request blocking rate.

3.1 Edge Bandwidth Broker (EBB)

This section describes how an EBB performs its admission control and bandwidth management functions. An EBB independently accepts or rejects new flows, releases unused resources back to CBB, and reallocates existing flows if current bandwidth allocations are changed by the CBB. Toward this end, EBB sometimes needs to perform local rerouting. In this section, we first describe how to make admission control upon a new request, then how to release unused resources, and finally how to reallocate existing flows triggered by the CBB request.

The EBB$_{s_i}$, the broker at ingress router s_i, maintains:

- for each existing flow f_{ij}, its assigned path p_j.
- for each LSP $p \in P_i$, its allocated bandwidth K_{ip}, residual bandwidth r_{ip} and associated cost t_{ip}.

Note that $r_{ip} = K_{ip} - \sum_{p_j = p} f_{ij}$.

When a new request f_{ij} arrives, EBB$_{s_i}$ can locally admit the request if $\exists p, r_{ip} \geq f_{ij}$, and then update its record of flow assignments and residual bandwidth accordingly. Note that there could be many strategies in prioritizing the choice among available LSPs. For example, it could be based on

the shortest path, the largest residual bandwidth, the smallest residual bandwidth, the flow instances in it with the earliest termination time, etc.

If the initial attempt for flow admission fails because none of the r_{ip} has sufficient bandwidth to meet the new request, EBB_{s_i} checks to see if local rerouting should be attempted. Local rerouting is performed if the total amount of residual bandwidth is larger than the new request, which implies that repacking could free up enough resource on some path to serve the request without signalling the CBB for additional bandwidth.

If local rerouting fails, or if the total residual bandwidth is less than the size requested, then EBB_{s_i} signals to the CBB to request for additional bandwidth for the flow class. If the CBB grants additional bandwidth, the pending flow instance will be admitted. Otherwise, EBB_{s_i} blocks the flow instance.

The EBB is also in charge of bandwidth management. To archive higher resource utilization and lower blocking rate, the EBB releases unused resources back to the CBB. It releases b_i amount of bandwidth if $\exists p, r_{ip} \geq b_i$, or $\sum_{p \in P_i} r_{ip} \geq b_i$ upon flow termination.

If the CBB send a message to an EBB to inform reallocation of some reserved resources due to global rerouting, the EBB runs local rerouting to comply with the request and replies to the CBB with result.

3.1.1 Local Rerouting

In this section, we describe the generic local rerouting algorithm as well as the local rerouting algorithms used in experimental section 4.

Generic Local Rerouting Algorithm

1. ~~Select a set of candidate flows $V_i \subseteq F_i$~~ for rerouting;
2. Compute the available bandwidth $vr_{ip} = r_{ip} + \sum_{f_{ij} \in V_i, p_j = p} f_{ij}$ for each LSP $p \in P_i$, as if flows in V_i do not exist;
3. Reroute all flows in V_i to use up the set of available bandwidth vr_{ip};
4. Attempt to admit the new request after rerouting is completed for flow admission.

The first three steps are general and the last one is for flow admission only. If the algorithm fails to reroute any of the flows in V_i in Step 3, it aborts the attempts and rollbacks all the flows to their original placements.

A wide range of algorithms are possible in Step 1 and 3. For example, the candidate set V_i can be the set F_i, meaning all flows are candidates for rerouting. The set V_i could contain just enough flows such that $\sum_{f_{ij} \in V_i} f_{ij}$ is greater then the new request for flow admission. In Step 3 one could reroute the flows in the lowest cost path first, in the highest cost path first, or to keep as many flows in their current path as possible. A much simpler and efficient alternative would be to only reroute flows in the highest cost path.

EBBs can activate local rerouting in many occasions. In addition to the scenarios just mentioned in the previous section, an EBB can reroute flow instances upon termination of existing instances, periodically, time-out based, or some combinations of them, similar to the list of alternatives studied in [10].

Following rerouting algorithm is used in the experimental section when a new request comes and we need to run local rerouting to repack existing flow. We first choose the flows using the path which has the highest residual bandwidth. Then we sort the flows in descendent order of bandwidth size and start with the highest bandwidth. Finally, we terminate the local rerouting algorithm if we crate enough capacity to accept the request.

Local rerouting on flow departure is similar to local rerouting for flow arrival except that instead of accepting a new flow, the goal is to free up a bandwidth bulk of b_i. If the EBB is able to reroute flows such that b_{ip} can be removed from a path p, the bandwidth is released back to the CBB. The test for whether such rerouting should be performed is decided by whether $\sum_{p \in P_i} r_{ip} \geq b_i$.

Finally, local rerouting can also be activated if global rerouting has been performed. In such a scenario, the CBB notifies all th EBBs whose LSPs' reserved capacity has been changed by the CBB. Upon receiving the reallocation message, the EBB_{s_i} runs local rerouting. Let path p be the one whose reserved capacity is decreased and path q be the one either newly set up or whose reserved capacity is increased. The EBB_{s_i} chooses the all flows in the path p and sorts them in descendent order in terms of bandwidth. It reroutes the flows until $\sum_{f_{ij} \in p} f_{ij} \leq K_{ip}$. After the local rerouting, the EBB_{s_i} sends a message to the CBB to notify the result.

3.2 Central Bandwidth Broker (CBB)

The CBB is responsible for allocating aggregated bandwidth along LSPs to each flow class. Upon receiving a request b_i from EBB_{s_i}, the CBB runs a resource allocation algorithm to check if any path $p \in P_i$ has sufficient residual bandwidth. If the allocation is successful, the CBB "admits" the request, returns b_{ip} to EBB_{s_i}, and updates the residual bandwidth of the links along the path p accordingly. On the other hand, should the allocation fail, the CBB can choose to resize existing LSP in order to free up resources to serve the new bulk request. The motivation behind global rerouting is to allow more efficient usage of resources by moving flows that have been routed on higher cost paths in the past.

Note that the global rerouting algorithm need not be triggered at the time of a bandwidth request. The CBB can take a more proactive action to run the algorithm before congestion occurs. The CBB can also run the algorithm periodically, or when a resource amount is returned by an EBB.

In order to perform its duties, the CBB maintains

- For each flow class FC_i, the set of bulks $B_i = \{b_{ip}\}$, i.e., the bulks of bandwidth allocated to the class FC_i along each path $p \in P_i$, and the associated cost t_{ip} of each bulk;

- for each link l, the initial capacity C_l and the residual bandwidth r_l.

Let $p \wedge q \neq \emptyset$ mean two paths p and q share at least a link. Otherwise, $p \wedge q = \emptyset$. Let $B = \bigcup_i B_i$. Below, we give generic description of the global rerouting algorithm.

Generic Global Rerouting Algorithm

1. Selects a set $W \subseteq B$ as the set of targeted bulks for rerouting, or $W = \{b_i\}$ for accepting new requests;
2. For each $b_{ip} \in W$, do
   (a) Identify a set of candidate paths $P_i' \subseteq (P_i - \{p\})$, where each $p' \in P_i'$ is potentially a new path for b_{ip};
   (b) Iterate through each $p' \in P_i'$
      i. Identify the set of bulks $Z = \{b_{jq} \mid b_{jq} \in B, q \wedge p' \neq \emptyset\}$;
      ii. Reroute enough bulks in Z such that there is enough residual bandwidth in path p' for the b_{ip};
      iii. Try the next path in P_i' if residual bandwidth $r_{ip'} < b_{ip}$, i.e. path p' is not feasible;
   (c) If every path $p' \in P_i'$ is not feasible, b_{ip} cannot be rerouted;
3. If any $b_{ip} \in W$ cannot be rerouted, the algorithm terminates and all bulks remain at their original paths. Otherwise, for any of the bulks which have been routed over a different path, the CBB notifies the corresponding EBB and waits for response. If the CBB gets positive responses from the notified EBBs, it updates its states. If the global rerouting algorithm is run for a new request, after getting positive responses, the CBB accepts the new request, notifies the EBB_{s_i}, and updates its states.

There could be many variations of the global rerouting algorithm. The selection of the rerouting set W, the candidate path set P_i' could determine how aggressive one intends to reroute. In addition, the choices of the bulk set Z and the rerouting of elements in Z, or even how one steps through the paths in P_i', all can affect the outcome of global rerouting.

As an extreme case, W can be equal to $\bigcup_i B_i$, that is, rerouting all existing bulk allocations. In such a case, one can solve the global rerouting problem by a multi-commodities flow placement algorithm which maximizes the minimum residual bandwidth on all links. The algorithm solves for all aggregates at the same time instead of breaking the computation into multiple iterations.

Note that when a bulk request b_i triggers the global rerouting, one can simply let $W = \{b_i\}$ and follow the steps.

While many instantiations of the global rerouting exist, we are interested in those that are practical and efficient. Following is the global rerouting algorithm we evaluate in the experimental section. Global rerouting takes place only when a new bulk request b_i cannot be satisfied based on the current link residual bandwidth $\{r_l\}$. Hence, W contains only the bulk b_i, which does not have an existing path p. We assume that all bandwidth aggregates have the *same size*. Let cost t_{ip} be determined by *hop count* of the path p.

Global Rerouting Algorithm Evaluated

1. Let P_i' be the set of lowest cost paths feasible to the flow class FC_i;
2. For each path $p' \in P_i'$, do
   (a) Remove temporarily b_i from the residual bandwidth of links along the path p', as if the new bulk request can take the path p'. As a result, congested links along p' have negative residual bandwidth;
   (b) Identify the set $Z = \{b_{jq} \mid b_{jq} \in B, p'$ and q share at least one congested link $\}$ and prioritizes the bulks in Z based on the cost t_{jq};
   (c) Starting from the highest cost bulk $b_{jq} \in Z$, reroute b_{jq} to an alternative path $q' \in P_j$ that has equal or lower cost than q and sufficient residual bandwidth using Widest Shortest Path (WSP) [4].
   (d) The iteration through bulks in Z terminates whenever links along the path p' all have non-negative residual bandwidth again, in such a case the global rerouting has succeeded and the CBB grants $b_{ip'}$ to EBB_{s_i}. CBB notifies corresponding EBBs whose bulks have been rerouted.
   (e) If some links along the path p' are still short of bandwidth after attempting to reroute bulks in Z, b_i is added back to the links along p'. Go back to Step 2;
3. The paths in P_i' have been exhausted and the global rerouting fails. The CBB rejects b_i.

We use following notation for resource allocation schemes: 1) No-Rerouting Algorithm (NRA), 2) Local Rerouting Algorithm (LRA) only, 3) Global Rerouting Algorithm (GRA) only, 4) Global+Local Rerouting Algorithm $(GLRA)$, and 5) Optimum Algorithm: In this case, we choose all existing flows for local rerouting, and all existing bulks and the new bulk request for global rerouting. Optimum algorithm is equivalent to a multi-commodities flow placement algorithm. This algorithm is used as a benchmark for the above 4 algorithms in terms of blocking rate.

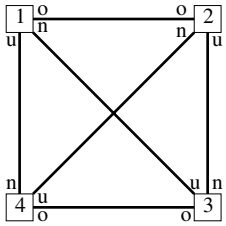
4 Experimental Investigation

In this section, we evaluate the various resource allocation schemes described in Section 3. The local and global rerouting algorithms evaluated are described in Section 3.1.1 and 3.2. We used two different network topologies in our experiments. The first topology is a 4 nodes, fully connected mesh network shown in Figure 2. The second topology is an irregular network shown in Figure 8 [5].

In the simulation, we assume that flows arrive as a poison process with rate λ and have exponential holding time with mean $1/\mu$. Network load ρ is defined as $\frac{\lambda}{\mu \cdot C}$ where C is each link capacity. Each data point is obtained with the average of 5 runs and to ensure that the comparisons are consistent, the same 5 random seeds are used for each point for every algorithm. We assume the underlying routing algorithm is the Widest Shortest Path (WSP) [4].

The two parameters of interest in the experiments are *blocking rate* and *routing cost*. Routing cost is measured as the sum of two components: bandwidth request sent from the EBBs to the CBB and rerouting computation performed by the CBB. We exclude the local rerouting performed by EBBs from routing cost because it does not affect the scalability of the architecture. For simplicity, we assign a cost of 1 unit to each bandwidth request and rerouting computation. We believe this assignment is sufficient for the purpose of studying the trade-off between blocking rate and routing cost. Lastly, for ease of presentation, all routing cost are normalized such that the routing cost of the *No-Rerouting Algorithm (NRA)* with bandwidth allocation size b_i = 1 is set to 1.0.

4.1 Experiment Set 1: Mesh Network



o : over-loaded
n : normal-loaded
u : under-loaded

Figure 2. Mesh Topology

In this section, we evaluate the two-tier resource allocation framework with fully connected mesh topology. We set all link capacities to 64 units and we vary bandwidth allocation size in discrete sizes of 1, 2, 4, 8 and 16. We set each class load to be the same and change network load from 0.7 to 1.2 for each flow class. We set $\lambda = \rho/10$ arrival per sec, $\mu = 1/640$ departure per sec, and simulation time 3 hours. At the end of 3 hours, approximately 1000 flows per class will leave the system when $\rho = 1$. First, we assume homogeneous flow rate and set the flow rate 1 unit of capacity.

Aggregated bandwidth control with no rerouting serves as the baseline for the other algorithms and the results are presented in Figure 3. The plot in Figure 3 shows how blocking rate varies with network load for different bandwidth allocation size. When network load is low (< 0.85), increasing bandwidth allocation size from 1 to 16 increases the blocking rate. However, as the network load increases, the reverse is true. Large bandwidth allocation size reduces blocking rate. This result can be explained by the fact that

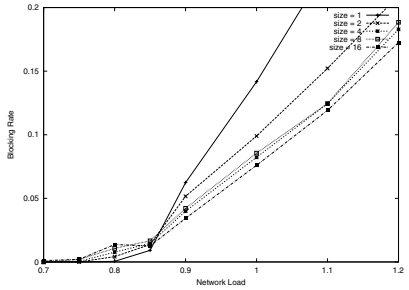


Figure 3. Blocking rate for *No Routing* with different bandwidth allocation size

at a higher load, a larger bandwidth allocation size allows a flow class to seize a significant portion of the cheapest (minimum hop) path quickly. Trunk reservation on the *primary link* (minimum hop) has been used as a way to limit excess alternate path calls from reducing the networking throughput to a very low level [9].

LRA performs better than or same as NRA for every bandwidth allocation size in terms of blocking rate. However, this gain comes at the expense of higher routing cost. However, note that the cross over in blocking rate as network load increases is not observed for LRA. This is due to the fact that local rerouting removes expensive paths, thus ensuring that a smaller bandwidth allocation size is more efficient. We omit the figure due to space limit. GRA performs better than or same as NRA for every bandwidth allocation size in terms of blocking rate, but slightly worse than LRA. Again, the reduction in blocking rate comes at the expense of higher routing cost.

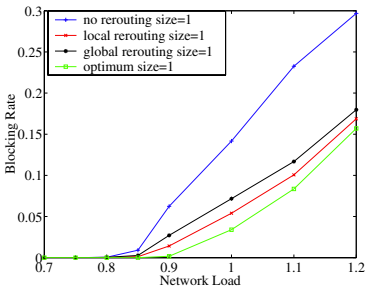


Figure 4. Blocking rate for different algorithms with allocation size = 1

The algorithms are compared for the fixed bandwidth allocation size (b_i) of 1 and 8 to illustrate the range of possible behaviors. Figure 4 shows the result for allocation size 1 and Figure 5 shows the result for allocation size 8.

The case for b_i = 1 is equivalent to flow based routing for NRA. The blocking rate reduction in LRA and GRA comes solely from rerouting. From Figure 4, the improvement of LRA and GRA over NRA is fairly substantial for a wide range of traffic load (0.85 - 1.2). It is interesting to note from Figure 4, that LRA with very small computation overhead at the edge routers performs so well the optimal algorithm is

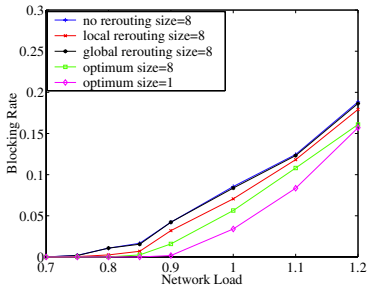


Figure 5. Blocking rate for different algorithms with allocation size = 8

only modestly better. When b_i =8, the reduction in blocking rate becomes significantly smaller. In fact, blocking rate for GRA is very similar to NRA. However, even the optimum algorithm can not reduce blocking rate significantly for bulk size 8.

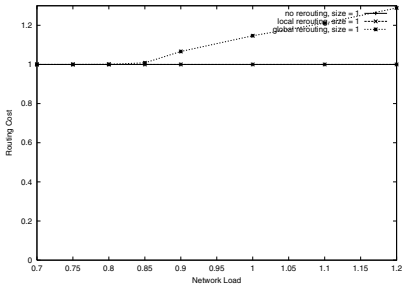


Figure 6. Routing cost for different algorithms with allocation size = 1

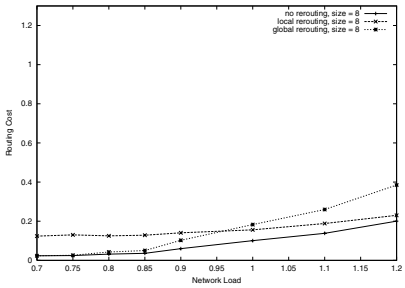


Figure 7. Routing cost for different algorithms with allocation size = 8

Figure 6 shows the routing cost for allocation size 1 and Figure 7 shows for allocation size 8. Routing cost for LRA is larger than or equal to NRA for all allocation sizes because LRA is often too aggressive in returning unused bandwidth. We would like to note that the routing cost for LRA is pure signalling cost. We see that routing cost for GRA is significantly lower than LRA and is much closer to NRA for small allocation size and lower load. The reason for this is that LRA returns unused resources as long as it could. Therefore, LRA requires more signalling. On the other hand, we do not perform global rerouting for lower network load because the blocking rate is very small. For bigger allocation size and higher network load, the routing cost for GRA is higher than LRA because GRA attempts global rerouting too often.

For brevity, the result of global + local rerouting algorithm (GLRA) will not be shown here. The blocking rate of GLRA is similar to LRA with slightly higher routing cost.

We remove homogeneous flow rate assumption. We choose flow request size randomly from uniform distribution between 0.5 and 1.5. We compare local rerouting algorithm with no rerouting algorithm when allocation size is 8. In this case, we find that simple local rerouting can reduce both blocking rate and routing cost significantly. We omit the figure due to space limit.

We also explore cases where each flow class has different loads. For this case, we define 3 different traffic loads as over-load (o), under-load (u) and normal-load (n). Figure 8(a) shows the load classification on the various (unidirectional) links. We choose a load of 0.8 as the normal load and tried 6 different sets of load distributions. These load distributions are given in Table 4.1 and are listed in increasing order of how unbalanced the loads are.

Set #	1	2	3	4	5	6
Over-Load	0.8	0.85	0.9	1.0	1.1	1.2
Normal-Load	0.8	0.8	0.8	0.8	0.8	0.8
Under-Load	0.8	0.75	0.7	0.6	0.5	0.4

Table 1. Mesh Loads for Special Case

The results in Set 1 are essentially the balance load measurement with load = 0.8 from Section 4.1. The performance of GLRA tends to trail that of LRA in most experiments but for fairly unbalanced load such as set 6, GLRA performs better than LRA and has the best tradeoff. This can be explained by noting that with a highly unbalanced load, GLRA is able to better utilize more alternative paths in addition to local rerouting. We omit the figure due to space limit.

4.2 Experiment Set 2: Irregular Topology

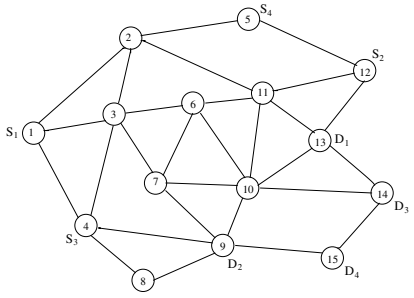


Figure 8. Irregular Topology

In this section, we investigate the effect of rerouting on an irregular topology as shown in Figure 8 (b). All link capacities are 64 units. Four source destination pairs are

defined as $((S_1, D_1), (S_2, D_2), (S_3, D_3), (S_4, D_4))$. The loading on each pair is uniform, and varies from 1.0 to 2.0.

In order to illustrate how the topology affects rerouting, we list the number of alternative paths for the 4 source destination pairs up to 5 hops. The direct path for (S_1, D_1) has 3 hops. There are 8 4-hop paths and 25 5-hop paths. For (S_2, D_2), there are 2 3-hop paths, 8 4-hop paths and 15 5-hop paths. For (S_3, D_3), there are 2 3-hop paths, 6 4-hop paths and 19 5-hop paths. Finally, for (S_4, D_4), there are a 4-hop path and 11 5-hop paths.

All experiments performed for the balance-load over mesh topology are also performed for the irregular topology. The results are similar except that the reduction of blocking rate is not as significant as in the mesh topology. We find that for lower network load, the gain is less significant due to the availability of large numbers of alternate paths. The result shows that for this topology, all three rerouting schemes perform well relative to NRA. While routing cost is high for allocation size of 1, for allocation size of 2, 4 and 8, both the blocking rate and routing cost are significantly lower than the baseline scenario of flow based routing.

For allocation size of 1,2,4 and 8, the blocking rate remains fairly constant while routing cost decreases. This is a result of the large number of alternative paths available for all 4 traffic pairs. Therefore, even when a large amount of resources have been allocated, enough resources remains such that the traffic from different source-destination do not interfere. We omit the figures due to space limit.

5 Conclusion

This paper has presented a two-tier resource allocation model with generic local and global rerouting algorithms. Based on the generic model, specific implementations of the local and global rerouting algorithms are presented and evaluated. In the evaluation, we compared three rerouting schemes with the the baseline algorithm which performs aggregate resource allocation with no rerouting. Our results show that rerouting improves performance over a wide range of network loads on two different network topologies. In particular, we show that depending on the network load and topology, it is possible to reduce both blocking rate and routing cost.

Obviously, there are many more issues that require further study. We plan to investigate the effect of variable request size and the use of variable bandwidth allocation size. Also, we feel that it is important to understand the relationship between topologies and rerouting performance. It will be very useful if given any network, the benefits of rerouting can be quantified.

References

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

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

[3] R. Cohen. Smooth intentional rerouting and its applications in atm networks. In *Proceedings of IEEE Infocom'94*, June 1996.

[4] R. Guerin, A. Orda, and D. Williams. Qos routing mechanisms and ospf extensions. In *Proceedings of 2nd IEEE Global Internet Mini-Conference*, November 1997.

[5] M. S. Kodialam and T. V. Lakshman. Dynamic routing of bandwidth guaranteed tunnels with restoration. In *Proceedings of IEEE INFOCOM 2000*, March 2000.

[6] T. Li and Y. Rekhter. A provider architecture for differentiated services and traffic engineering (PASTE). RFC 2430, IETF, October 1998.

[7] S. Ohta and K. Sato. Dynamic bandwidth control of the virtual path in an asynchronous transfer mode network. *IEEE Transactions on Communications*, 40(7):1239 – 1247, July 1992.

[8] A. Orda, G. Pacifici, and D. E. Pendarakis. An adaptive virtual path allocation policy for broadband networks. In *Proceedings of Infocom'96*, June 1996.

[9] K.W. Ross. *Multiservice Loss Model for Broadband Telecommunication Networks*. Springer, 1995.

[10] Eric W. M. Wong, Andy K. M. Chan, and Tak-Shing P. Yum. Analysis of rerouting in circuit-switched networks. *IEEE/ACM Transactions on Networking*, 8(3):419 – 427, June 2000.