

Using Rerouting to Improve Aggregate Based Resource Allocation¹

Ali E. Helvaci and Coskun Cetinkaya
 Wichita State University, Department of Electrical and Computer Engineering
 1845 Fairmount, Wichita, KS 67260-0044
 Emails: {aehelavi,coskun.cetinkaya}@wichita.edu

Mehmet B. Yildirim
 Wichita State University, Industrial and Manufacturing Engineering Department
 1845 Fairmount, Wichita, KS 67260-0035
 Email: bayram.yildirim@wichita.edu

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 used 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.

Index Terms—Resource Reservation, Re-routing, Bandwidth Broker

I. INTRODUCTION

Future integrated service networks will support multiple and diverse applications 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 [6]. Such algorithms achieve this goal by calculating a feasible path for each connection. When the number of service requests and the 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 a 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 signaling the CBB. Thus, the CBB is infrequently signaled to achieve scalability, yet without sacrificing 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 signaling and computation. On the other hand, if a class' aggregate demand oscillates quickly, we would have to choose to either rapidly readjust the reservation to track the demand (we need to signal frequently and lose the scalability), or leave a gap between the demand and the reservation (we encounter low utilization and high blocking 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

¹ An Earlier version of this paper appeared in Proceedings of IEEE ISCC 2003.

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.

In this paper, we extend our previous work [13] to introduce a new analytical model for obtaining optimal bulk size for given network constraints.

The paper is organized as follows. Section II presents related work. Section III describes the network architecture and presents the two-tier resource allocation framework and the rerouting algorithms. We present mathematical analysis for a single node in Section IV. Section V presents the experiment results and Section VI the conclusion.

II. 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 [9], 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 [10] the threshold based VP allocation scheme in [9] 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 [8] (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 [3], a set of requirements for traffic engineering over MPLS is presented. It identifies the functional

capabilities required to implement policies that facilitate efficient and reliable network operations in an MPLS domain. Other state reduction techniques also include those described in [4] for supporting Integrating Services over the DiffServ network.

In the world of 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 [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 an attempt is made 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 [12].

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 a circuit-switched network. We developed a two-tier resource allocation framework with two-level rerouting algorithms. The rerouting is performed on an aggregate basis, in addition to a per flow basis, in order to reduce signaling load and increase the network utilization. Finally, the algorithm is designed to work with any topology.

III. SYSTEM MODEL AND ALGORITHMS

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 [5].

The admission control of resource requests is based on a two-tier bandwidth brokers architecture, as depicted in Fig. 1. A *Central Bandwidth Broker (CBB)* provisions LSPs and allocates bandwidth along these 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*.

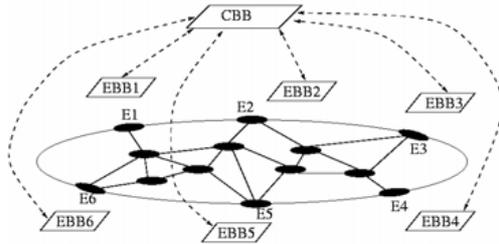


Figure 1. Architecture of two-tier bandwidth broker.

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 *EBBs* 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 , the *EBBs_i* chooses p such that $r_{ip} \geq f_{ij}$ and admits the request. If none of the LSPs that the *EBBs_i* manages on behalf of class FC_i has sufficient residual bandwidth, the *EBBs_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 the *EBBs_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, the *EBBs_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 *EBBs_i*, which in turn accumulates them together in K_{ip} . The *EBB* could release the bandwidth in the same or

different increments. In our study we assume that *EBBs_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 the *EBBs_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 the 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 done 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 unknown future traffic, we do not consider getting an optimal global solution.

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

A. 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 the *CBB*, and reallocates existing flows if current bandwidth allocations are changed by the *CBB*. Toward this end, the *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 *EBBs_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, the $EBBs_i$ 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, the $EBBs_i$ 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 resources on some path to serve the request without signaling the CBB for additional bandwidth.

If local rerouting fails, or if the total residual bandwidth is less than the size requested, then the $EBBs_i$ 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, the $EBBs_i$ 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

$$\sum_{p \in P_i} r_{ip} \geq b_i \text{ upon flow termination.}$$

If the CBB send a message to an EBB to inform it of the need for 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 the result.

1) Local Rerouting

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

Generic Local Rerouting Algorithm

- 1) Select a set of candidate flows $V_i \subseteq F_i$ for rerouting;
- 2) Compute the available bandwidth $vr_{ip} = r_{ip} + \sum_{f_{ij} \in V_i, p_i=p} f_{ij}$ 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 than 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 could 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 the above, similar to the list of alternatives studied in [12]. Note that when all instances of the same flow class require the same amount of bandwidth, such as in the case of voice traffic, local rerouting on request arrival does not help in general, especially when each bulk size is a multiple of a flow size. However, local rerouting on flow departure can be beneficial.

The following rerouting algorithm is used in the experimental section when a new request comes and we need to run local rerouting to repack the 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 create 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 the $EBBs$ whose LSPs' reserved capacity has been changed by the CBB . Upon receiving the reallocation message, the $EBBs_i$ 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 $EBBs_i$ chooses all the flows in 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 $EBBs_i$

sends a message to the CBB to notify it of the result.

B. Central Bandwidth Broker (CBB)

The *CBB* is responsible for allocating aggregated bandwidth along the LSPs to each flow class. Upon receiving a request b_i from an *EBBs_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 *EBBs_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 an existing LSP to free up resources to serve the new bulk request. The motivation for 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 $p \wedge q \neq \phi$ mean two paths p and q share at least one link. Otherwise, $p \wedge q = \phi$. Let $B = \cup_i B_i$. Below, we give a generic description of the global rerouting algorithm.

Generic Global Rerouting Algorithm

- 1) Select 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 $Z = \{b_{jq} \mid b_{jq} \subseteq B, q \wedge p' \neq \phi\}$;
 - 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 *EBBs* and waits for responses. 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 *EBBs_i*, and updates its states.

There could be many variations of the global rerouting algorithm. The selection of the rerouting set W and 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 $\cup_i B_i$, 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. The following is the global rerouting algorithm we evaluated 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 the *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 $Z = \{b_{jq} \mid b_{jq} \subseteq B, p' \text{ and } q \text{ share at least one congested link}\}$ and prioritizes the bulks in Z based on the cost t_{ip} ;
 - 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 the Widest Shortest Path (WSP) [6].
 - d) The iteration through bulks in Z terminates whenever links along the path p' all have non-negative residual bandwidths again in such a case the global rerouting has succeeded, and the *CBB* grants b_{ip} to the *EBBS*_{*i*}. The *CBB* notifies the 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 the 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. The 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.

IV. ANALYTICAL PERFORMANCE MODEL

In this section, we develop a simple model to capture the key elements of the performance of aggregation, namely we introduce a N-dimensional Markov model in which flows arrive as a Poisson process with rate λ and have exponential holding time with rate $1/\mu$. We describe a baseline scenario in which aggregate classes are multiplexed onto a backbone link.

A. Analysis Model

In this section, we map an Aggregate Based Resource Allocation scheme into a N-dimensional Markov model

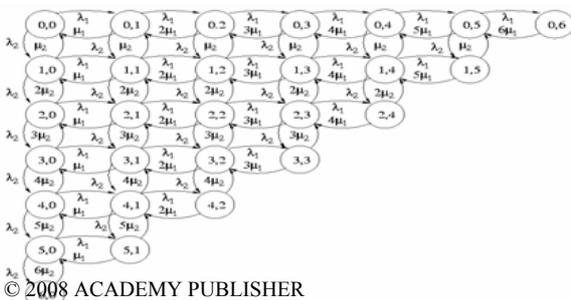


Figure 2 (a). IntServ with bulk size 1

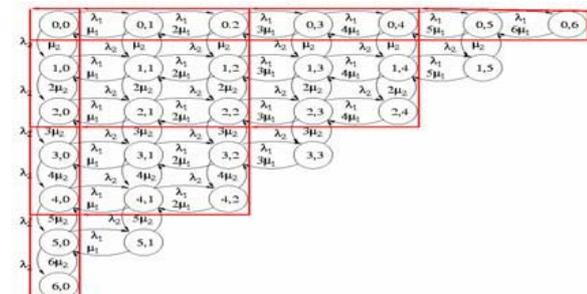
to investigate performance trade off. Since we focus on guaranteed service in this work, we have to make sure that the total reserved bandwidth remains less than or equal to link capacity, i.e., $\sum_{i=1}^N K_i \leq C$ where C is the total link capacity, K_i is the reserved bandwidth for i^{th} class, and N denotes the number of active classes. We assume that all flows are identical and each flow's rate is a unit of bandwidth. Therefore, C represents the maximum number of flows that can be accommodated in the link any given time. Furthermore, we assume that flows of class i arrive as a Poisson process with rate λ_i and have an exponential holding time with a mean of $1/\mu_i$. In the model, state vector $S(n_1, n_2, \dots, n_N)$ is defined to mean that at any instant there are n_i flows from class i . The Markov model helps us to calculate each class performance metrics such as class reserved bandwidth utilization, blocking probability, and signaling rate, defined in the following subsections. Our goal is to show the basic trade off when the Aggregate Based Resource Allocation scheme is applied. Therefore, we further assume that each of the class loads are also equal. Investigating different settings such as heterogeneous class load, flow rate, and bulk sizes, and what happens when the link capacity is not a multiple of bulk sizes etc. is out of this paper's scope.

In Fig. 2, we have shown a two-class state diagram for $C=6$. State $S(i, j)$ is defined to mean that at any instant there are i flows from class 1 and j flows from class 2. Fig. 2 (a) contains state diagram for the IntServ architecture ($b=1$) and (b) contains a diagram for Aggregate Based Resource Allocation with a bulk size of ($b = 2$) for both classes.

B. State Probability

In this section, we derive state probability with a bulk size vector \bar{b} . Let $\Pr_{\bar{b}}(i_1, i_2, \dots, i_N)$ be the statistical equilibrium joint probability that at any instant there are i_j customers from class j in the system where vector \bar{b} consists of each class bulk size b_j . For each class j , the class' arrival rate is λ_j , and its departure rate is μ_j . Using statistical equilibrium state equations, we can show that state probabilities are

$$\Pr_{\bar{b}}(i_1, \dots, i_N) = \frac{(\lambda_1 / \mu_1)^{i_1}}{i_1!} \dots \frac{(\lambda_N / \mu_N)^{i_N}}{i_N!} \Pr_{\bar{b}}(0, \dots, 0). (1)$$



(b) Aggregate based resource allocation with bulk size 2.

Since $\sum_{S_b} \Pr_b(i_1, \dots, i_N) = 1$, we can calculate

$\Pr_b(0, \dots, 0)$. For this reason, let us define

$$L_{i_k} = \left[\frac{C - \lceil i_1/b_1 \rceil \cdot b_1 - \dots - \lceil i_{k-1}/b_{k-1} \rceil \cdot b_{k-1}}{b_k} \right] \cdot b_k. \quad (2)$$

Therefore, we get

$$\Pr_b(0, \dots, 0) = \left(\sum_{i_1=0}^{L_{i_1}} \dots \sum_{i_N=0}^{L_{i_N}} \frac{(\lambda_1/\mu_1)^{i_1}}{i_1!} \dots \frac{(\lambda_N/\mu_N)^{i_N}}{i_N!} \right)^{-1}. \quad (3)$$

C. Network and Reserved Bandwidth Utilization

In this section, we investigate multi-class network utilization and individual class' reserved bandwidth utilization under Aggregate Based Resource Allocation.

Network utilization for bulk size vector \bar{b} is

$$U_{\bar{b}} = \sum_{i_1=0}^{L_{i_1}} \dots \sum_{i_N=0}^{L_{i_N}} \frac{\sum_{j=1}^N i_j}{C} \cdot \Pr_b(i_1, \dots, i_N). \quad (4)$$

Fig. 3 (a) depicts network utilization normalized by IntServ results for different bulk sizes $\{2,4,8,16,32\}$, and Fig. 3 (b) depicts network utilization for IntServ and $b=32$, when there are two classes, link capacity 64, and network load (0.5,1.5). As expected, network utilization degradation occurs when aggregation size increases. However, as shown in Fig. 3 (a), network utilization relative to IntServ decreases when network load increases up to some load, then it starts increasing. The reason behind this is that each class highly utilizes its reserved bandwidth for higher network loads.

Reserved bandwidth utilization with bulk size b_k for class k is

$$R_{b_k}^k = \sum_{i_1=0}^{L_{i_1}} \dots \sum_{i_N=0}^{L_{i_N}} \frac{i_k}{\lceil i_k/b_k \rceil \cdot b_k} \cdot \Pr_b(i_1, \dots, i_N). \quad (5)$$

Fig. 4 depicts the reserved Bandwidth utilization for the same network settings that was used Fig. 3. We find that Reserved bandwidth exponentially increases with network load, but decreases with bulk size. As we discussed previously, we encounter degradation on network utilization because individual classes do not fully

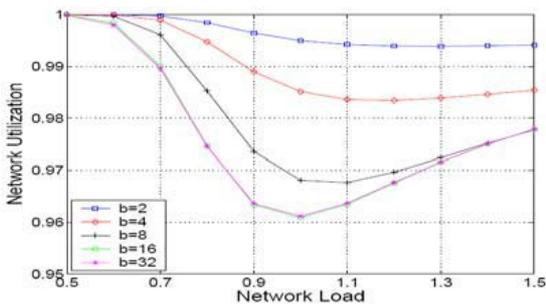


Figure 3 (a). Network utilization for different bulk size

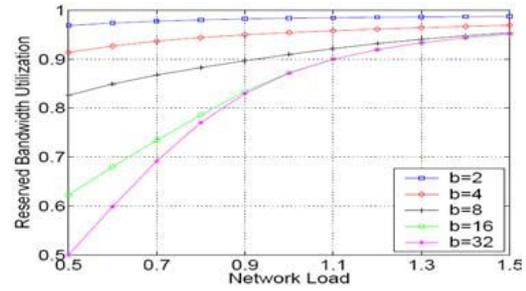


Figure 4. Reserved bandwidth utilization.

utilize their reserved bandwidth and unused capacity can not be used by other classes.

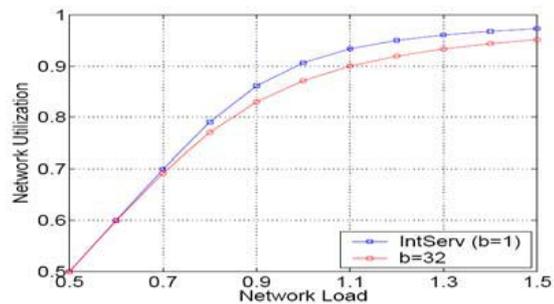
D. Signaling Rate

In this section, we investigate the effect of bulk size on signaling requirement. Signaling has two components due to arrival and departure. Let us define two sets of states in which we signal the network due to a new arrival or a departure, arrival set $\bar{A}_{b_k}^k$ and departure set $\bar{D}_{b_k}^k$. Arrival set $\bar{A}_{b_k}^k$ contains the states where a new arrival from class k requires a signal to the network for additional bulk, and departure set $\bar{D}_{b_k}^k$ contains the states where a departure from class k requires a signal to the network to release a bulk. For example, in Fig. 2 (b), $\bar{A}_{b_1}^1$ contains state $S(0,0)$, $S(0,1)$, ..., $S(0,6)$, $S(2,0)$, $S(2,1)$, ..., $S(2,4)$, $S(4,0)$, $S(4,1)$, $S(4,2)$, and $S(6,0)$. Let SR^k be the signaling rate of class k which is

$$SR^k = \sum_{S(i_1, \dots, i_N) \in \bar{A}_{b_k}^k} \Pr_b(i_1, \dots, i_N) \cdot \lambda_k + \sum_{S(i_1, \dots, i_N) \in \bar{D}_{b_k}^k} \Pr_b(i_1, \dots, i_N) \cdot \mu_k. \quad (6)$$

Therefore, total signaling SR is $SR = \sum_{i=1}^N SR^i$.

Fig.5 depicts the signaling rate normalized by the IntServ results for the same network setting that was used in Fig. 3. As shown in Fig. 5, signaling reduction remains almost constant over a small network load and starts decreasing with network load. The reason behind this is that for congested network, individual classes highly utilize their reserved bandwidth and need additional bandwidth with new arrivals. Therefore, class starts



(b) IntServ vs. Aggregate based resource allocation with $b=32$.

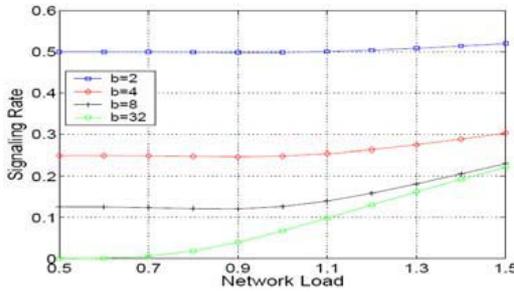


Figure 5. Signaling ratio.

signaling per flow bases. Hence, signaling reduction relative to IntServ decreases.

E. Flow Blocking Probability

In the previous section, we have shown that Aggregate Based Resource Allocation significantly reduces the signaling rate with small sacrifices in terms of the network utilization. Another disadvantage of Aggregate Based Resource Allocation is that we may encounter more blocking rate than in IntServ because we could not accept some upcoming flows although there is enough capacity to accept them. For example, let us assume we are at state $S(1,4)$ in Fig. 2 (b), and a class 2 flow wants to come into the network. Although we can support one more flow, we reject the flow because we can not give class 2 additional capacity. In this section, we calculate blocking probability under Aggregate Based Resource Allocation.

Let us define boundary state space \tilde{S}_b as states in which we can not accept one more flow for at least one of the classes. In Fig. 2 (b), we have boundary state as $\tilde{S}_b = \{S(0,6), S(0,5), S(1,4), \dots, S(6,0), S(5,0)\}$.

Let us also define the class-boundary state space for class k , $\tilde{S}_{b_k}^k$ as states in which we can not accept one more flow from class k . In Fig. 2 (b), we have the class-boundary state for class 1 as $\tilde{S}_{b_1}^1 = \{S(6,0), S(4,1), S(4,2), S(2,3), S(2,4), S(0,5), S(0,6)\}$. Now we can calculate the blocking probability B_b and the class blocking probability $B_{b_k}^k$ as

$$B_b = \sum_{S(i_1, \dots, i_N) \in \tilde{S}_b} P_{\mathbb{F}}(i_1, \dots, i_N), \quad (7)$$

$$B_{b_k}^k = \sum_{S(i_1, \dots, i_N) \in \tilde{S}_{b_k}^k} P_{\mathbb{F}}(i_1, \dots, i_N). \quad (8)$$

Fig. 6 depicts the blocking probability for the same network setting that was used in Fig. 3. As shown in Fig. 6, blocking probability increases with bulk size and the network load. However, for small load, Aggregate Based Resource Allocation introduces small additional blocking because the network load is not highly utilized all the time. When the network load increases, Aggregate Based Resource Allocation causes higher blocking probability because the unused capacity from a class can not be

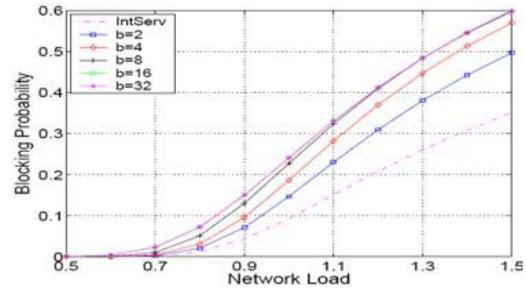


Figure 6. Blocking probability.

distributed to other classes. As shown in Section V, rerouting algorithms, introduced in Section III, further reduce blocking probability due to the Aggregate Based Resource Allocation scheme.

V. EXPERIMENTAL INVESTIGATION

In this section, we evaluate the various resource allocation schemes described in Section III. The local and global rerouting algorithms evaluated are described in Section III.A and III.B, respectively. We used two different network topologies in our experiments. The first topology is a 4-node, fully-connected mesh network shown in Fig. 7. The second topology is an irregular network shown in Fig. 15, as in [7].

In the simulation, we assume that flows arrive as a Poisson process with rate λ and have exponential holding time with a mean of $1/\mu$. Network load ρ is defined as

$$\rho = \frac{\lambda}{\mu \cdot C} \quad \text{where } C \text{ is each link capacity. Each data}$$

point is obtained using an average of 10 runs, and, to ensure that the comparisons are consistent, the same 10 random seeds are used for each point for every algorithm. We assume the underlying routing algorithm is the Widest Shortest Path (WSP) [6].

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 requests sent from the *EBBs* to the *CBB* and rerouting computation performed by the *CBB*. We exclude the local rerouting performed by the *EBBs* from routing cost because it does not affect the scalability of the architecture (we believe that the bottleneck is the *CBB*). 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 costs 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.

A. Experiment Set 1: Mesh Network

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 the network load from 0.7 to 1.2 for each flow

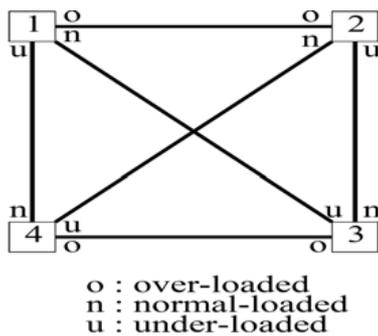


Figure 7. Mesh topology.

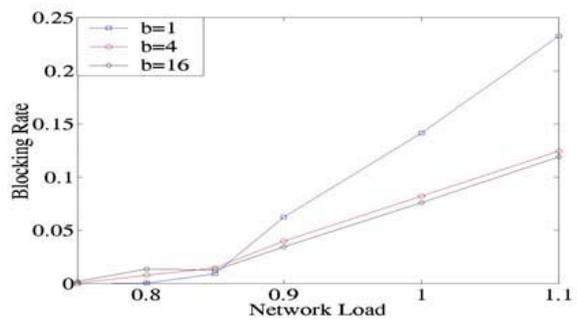


Figure 8. Blocking rate for *No-Rerouting* with different bulk size.

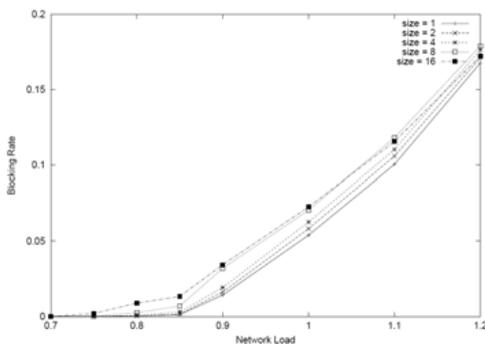
class. We set $\lambda = \rho/10$ arrival per sec, $\mu=1/640$ departure per sec, and the simulation time 3 hours. First, we assume a homogeneous flow rate and set the flow rate at 1 unit of capacity.

Aggregated bandwidth control with no rerouting serves as the baseline for the other algorithms, and the results are presented in Fig. 8. Fig. 8 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. A large bandwidth allocation size reduces the blocking rate. This result can be explained by the fact that 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

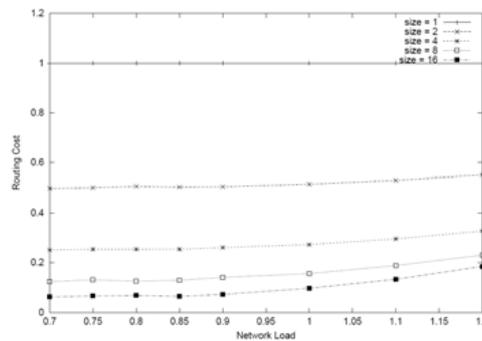
calls from reducing the networking throughput to a very low level [11].

LRA performs better than, or the same as, *NRA* for every bandwidth allocation size in terms of blocking rate. However, this gain comes at the expense of higher routing cost as shown in Fig. 9. 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. *GRA* performs better than, or the 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.

The algorithms are compared for the fixed bandwidth allocation size (b_i) of 1 and 8 to illustrate the range of possible behaviors. Fig. 10 (a) shows the result for allocation size 1 and (b) shows the result for allocation

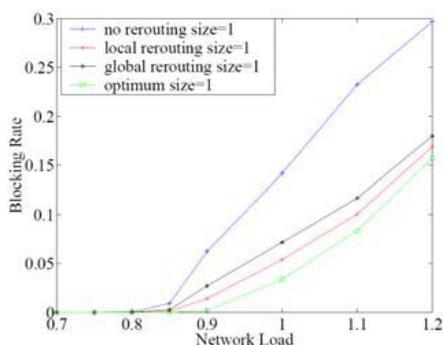


(a) Blocking rate

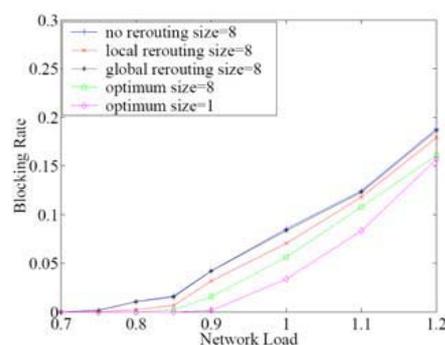


(b) Routing cost

Figure 9. Blocking rate and routing cost for Local Rerouting with different bulk size.

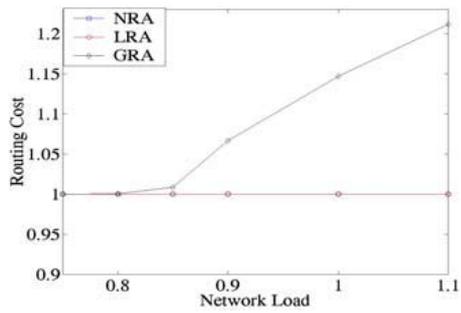


(a) Allocation size 1

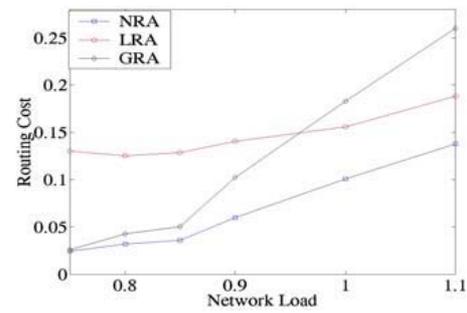


(b) Allocation size 8

Figure 10. Blocking rate for different algorithms.



(a) Allocation size 1



(b) Allocation size 8

Figure 11. Routing cost for different algorithms.

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 Fig. 10, 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 Fig. 10, that *LRA*, with a very small computation overhead at the edge routers, performs so well the optimal algorithm is only modestly better. When $b_i = 8$, the reduction in blocking rate becomes significantly smaller. In fact, the blocking rate for *GRA* is very similar to *NRA*. However, even the optimum algorithm can not reduce the blocking rate significantly for bulk size 8.

Fig. 11 (a) shows the routing cost for allocation size 1, and (b) shows the cost for allocation size 8. The 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 signaling cost. We see that routing cost for *GRA* is significantly lower than it is for *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 can. Therefore, *LRA* requires more signaling. On the other hand, we do not perform global rerouting for lower network loads because the blocking rate is very small. For bigger allocation sizes and higher network loads, the routing cost for *GRA* is higher than the cost for *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.

Fig. 12 presents one way to understand the trade-off between blocking rate and routing cost. The plot shows the relative performance of the various algorithms at the network load of 0.9 using different bandwidth allocation sizes with respect to the baseline scenario of *NRA* with $b_i = 1$ which takes the coordinate (0,0). A location of (-0.1, -0.5) means that the performance of the algorithm at this point has 10% less blocking, and 50% less routing cost than it does at the baseline scenario. Relative to the baseline, the lower left quadrant is the best scenario with less blocking and cost, and the upper right quadrant is the

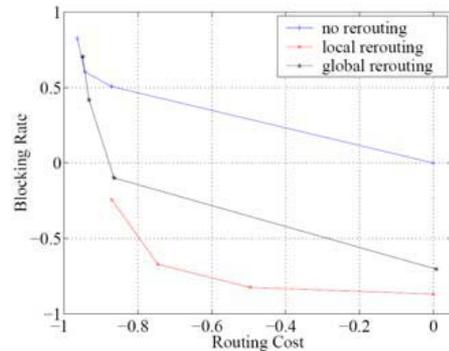


Figure 12. Trade-off between blocking rate and routing cost for balance load.

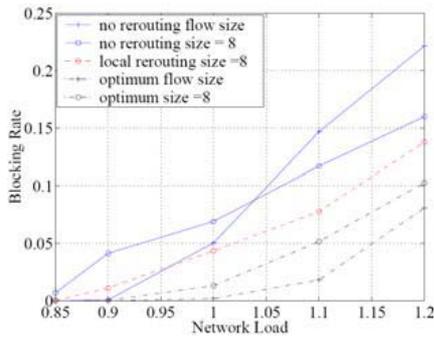
worst scenario with more blocking and cost. Each line in the figure corresponds to an algorithm with increasing allocation size (1, 2, 4, and 8) moving from right to left.

In Fig. 12, for *NRA*, increasing the allocation size increases the blocking rate while decreasing the cost. For *GRA*, the curve lies in the lower left quadrant only for $b_i = 2$. Finally, for *LRA*, all four points lie in the lower left quadrant. An allocation of size 4 ($b_i = 4$) provides the best trade-off between the blocking rate (reduced by 70%) and the routing cost (reduced by 75%). As seen in Fig. 12, we can reduce both the blocking rate and the routing cost with a simple local routing algorithm.

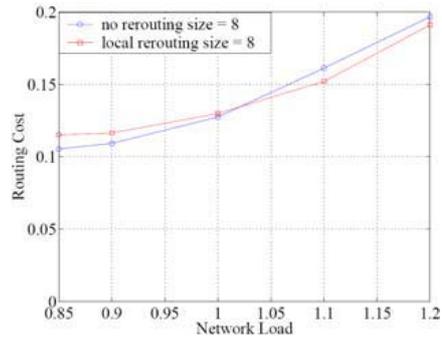
1) Heterogenous Flow and Class Rates

In this section, we remove the homogeneous flow rate assumption. We choose flow request size randomly from a uniform distribution between 0.5 and 1.5. We compare the Local Rerouting algorithm with the No Rerouting algorithm when allocation size is 8. In Fig. 13, we also include the No Rerouting algorithm when the allocation size is equal to the flow rate as well as the optimal algorithm with the allocation size equal to the flow rate and 8. In this case, we find that simple Local Rerouting can reduce both the blocking rate and the routing cost significantly.

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). Fig. 7 shows the load classification on the various (unidirectional) links. We choose a load of 0.8 as the



(a) Blocking rate



(b) Routing cost

Figure 13. Blocking rate and routing cost for heterogeneous flow rate with allocation size = 8.

TABLE I.
MESH LOADS FOR SPECIAL CASE

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

normal load and tried 6 different sets of load distributions. These load distributions are given in Table I and are listed in increasing order of how unbalanced the loads are.

The results in Set 1 are essentially the same as the balance load measurement with load = 0.8 from Section V.A. We will not show most of the plots in this set of experiments except for the figure on trade-off between blocking and cost. In Fig. 14, the plot shows the result for set 6 where the load distribution is fairly unbalanced, going from a load of 0.4 on the under-load links to 1.2 on the overload links. The results are similar to the results in Fig. 12 with the exception of *GLRA*. The performance of *GLRA* tends to trail that of *LRA* in most experiments but in this case, *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.

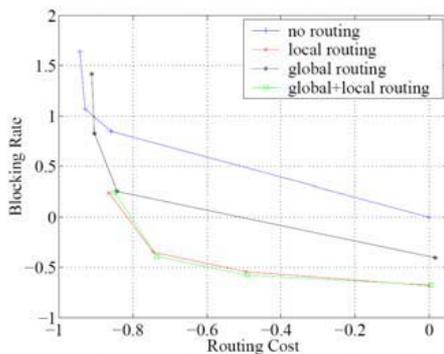


Figure 14. Trade-off between blocking rate and routing cost for unbalance load.

B. Experiment Set 2: Irregular Topology

In this section, we investigate the effect of rerouting on an irregular topology as shown in Fig. 15. 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 three hops. There are eight 4-hop paths and twenty five 5-hop paths. For (S_2, D_2) , there are two 3-hop paths, eight 4-hop paths and fifteen 5-hop paths. For (S_3, D_3) , there are two 3-hop paths, six 4-hop paths and nineteen 5-hop paths. Finally, for (S_4, D_4) , there are one 4-hop path and eleven 5-hop paths.

All experiments performed for the balance-load over mesh topology were also performed for the irregular topology. The results are similar except that the reduction of the blocking rate is not as significant as in the mesh topology. We find that for the 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 the allocation size of 1, for the allocation sizes of 2, 4, and 8, both the blocking rate and the routing cost are significantly lower than the baseline scenario of flow-based routing.

For allocation sizes of 1, 2, 4, and 8, the blocking rate

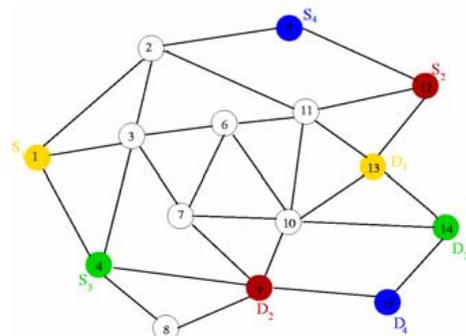


Figure 15. Irregular Topology.

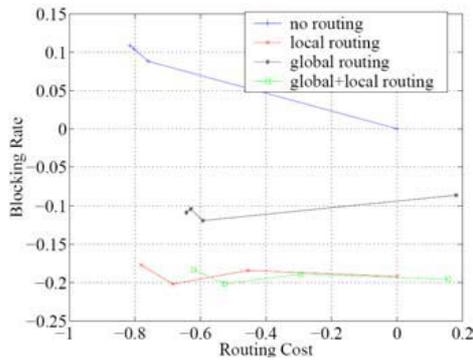


Figure 16. Trade-off between blocking rate and routing cost for irregular topology.

remains fairly constant while the 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 has been allocated, enough resources remain such that the traffic from different source-destination do not interfere as shown in Fig. 16.

VI. 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 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 the blocking rate and the routing cost.

ACKNOWLEDGMENT

The author would like to extend his thanks to Drs. Mun Choon Chan and Yow-Jian Lin for their guidance and insightful comments.

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*, vol. 6 (2), pp. 222–236, April 1998.
- [2] G. R. Ash, "Dynamic routing in telecommunications network," *McGraw Hill*, 1997.
- [3] D. Avduche et al., "Requirements for traffic engineering over MPLS," *RFC 2702, IETF*, September 1999.
- [4] Y. Bernet et al., "A framework for integrated services operation over diffserv network," *RFC 2998, IETF*, November 2000.
- [5] R. Cohen, "Smooth intentional rerouting and its applications in ATM networks," in *Proceedings of IEEE Infocom'94*, Toronto, CA, June 1996.
- [6] R. Guerin, A. Orda, and D. Williams, "Qos routing mechanisms and OSPF extensions," in *Proceedings of 2nd*

IEEE Global Internet Mini-Conference, Phoenix, AZ November 1997.

- [7] M. S. Kodialam and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," in *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
- [8] T. Li and Y. Rekhter, "A provider architecture for differentiated services and traffic engineering (PASTE)," *RFC 2430, IETF*, October 1998.
- [9] S. Ohta and K. Sato, "Dynamic bandwidth control of the virtual path in an asynchronous transfer mode network," *IEEE Transactions on Communications*, vol. 40 (7), pp. 1239 – 1247, July, 1992.
- [10] A. Orda, G. Pacifici, and D. E. Pendarakis, "An adaptive virtual path allocation policy for broadband networks," in *Proceedings of Infocom'96*, San Francisco, CA, June 1996.
- [11] K.W. Ross, "Multiservice loss model for broadband telecommunication networks," *Springer*, 1995.
- [12] E. W. M. Wong, A. K. M. Chan, and T. S. P. Yum, "Analysis of rerouting in circuit-switched networks," *IEEE/ACM Transactions on Networking*, vol. 8 (3), pp. 419 - 427, June 2000.
- [13] C. Cetinkaya, M.C. Chan and Y.J. Lin, "Aggregate Based Resource Allocation with Rerouting," in *Proceedings of IEEE ISCC'2003*, Kamer, Turkey, June 2003.

Ali Ersin Helavci received his B.S. degree from Istanbul University, Turkey in 2005 and is pursuing his MS degree in Electrical and Computer Engineering at Wichita State University. His research interests are computer networks, wireless ad hoc and mesh networks.

Coskun Cetinkaya received his B.S. degree from Anatolia University, Turkey in 1994, M.S. from the University of Southern California in 1998 and Ph.D. from Rice University in 2003. Since then, he has been at Wichita State University. His research interests are computer networks, wireless ad hoc and mesh networks, specifically Admission Control, Quality of Service, Medium Access Control (MAC) Protocols, Transport Protocols, and Scheduling Algorithms.

Mehmet Bayram Yildirim received his B.S. degree from Bogazici University, Istanbul, Turkey and M.Sc. degree from Bilkent University, Ankara, Turkey in Industrial Engineering in 1994 and 1996, respectively. He received his PhD degree from Department of Industrial and System Engineering in University of Florida in 2001. He is an assistant professor in the Industrial and Manufacturing Engineering Department at Wichita State University. His research interests are optimization, scheduling, logistics and supply chain, sustainable and green manufacturing.