End-to-End Quality of Service for High-End Applications

Ian Foster[*,†] Alain Roy[†] Volker Sander[*,‡] Linda Winkler[*]

Abstract

High-end networked applications such as distance visualization, distributed data analysis, and advanced collaborative environments have demanding quality of service (QoS) requirements. Particular challenges include concurrent flows with different QoS specifications, high bandwidth flows, application-level monitoring and control, and end-to-end QoS across networks and other devices. We describe a QoS architecture and implementation that together help to address these challenges. The Globus Architecture for Reservation and Allocation (GARA) supports flow-specific QoS specification, immediate and advance reservation, and online monitoring and control of both individual resources and heterogeneous resource ensembles. Mechanisms provided by the Globus toolkit are used to address resource discovery and security issues when resources span multiple administrative domains. Our prototype GARA implementation builds on differentiated service mechanisms to enable the coordinated management of two distinct flow types—foreground media flows and background bulk transfers—as well as the co-reservation of networks, CPUs, and storage systems. We present results obtained on a wide area differentiated services testbed that demonstrate our ability to deliver QoS for realistic flows.

[*]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.
[†]Department of Computer Science, The University of Chicago, Chicago, IL 60637, U.S.A.
[‡]Central Institute for Applied Mathematics, Forschungszentrum Juelich GmbH, 52425 Juelich, Germany

1 Introduction

Investigations of network quality of service (QoS) have tended to focus on the aggregration of relatively low-bandwidth flows associated with Web and media streaming applications. Yet the QoS requirements associated with these flows are not representative of all interesting applications. For example, distance visualization applications encountered in science and engineering can involve data transfers and media streaming at hundreds (ultimately thousands) of megabits per second (Mb/s), while the bulk data transfer operations required for replication or analysis of large datasets can require sustained high bandwidths expressed in terms of Terabytes per hour. Advanced collaborative environments can require complex mixes of these and other flows, with varying service level requirements and many interdependencies.

The development of QoS services for such high-end applications introduces major challenges at both the architecture and network levels. At the architecture level, new concepts and constructs are required for dealing with end-to-end flows that involve multiple scarce resources: for example, advance reservation mechanisms, to ensure availability [4, 5, 18]; co-reservation of network, compute, storage, and other resources [2]; control and monitoring APIs for application-level adaptation; and policy mechanisms able to deal with large reservations and complex hierarchical allocation strategies. At the network level, increasingly popular differentiated service (DS) mechanisms [1] introduce both opportunities and challenges. While DS has advantages in terms of scalability, it is not obvious whether and how it can support specialized high-end flows.

The work that we present in this article addressed both the architecture and network challenges just listed. We describe the Globus Architecture for Reservation and Allocation (GARA), a resource management architecture that builds on mechanisms provided by the Globus toolkit [6] to support secure immediate and advance co-reservation, online monitoring/control, and policy-driven management of a variety of resource types, including networks [8]. Then, we describe the application of GARA concepts and constructs to DS networks. We present a DS resource manager (i.e., bandwidth broker [1, 11]) and explain how this resource manager integrates with GARA facilities (e.g., advance reservation, authentication/authorization). We describe how this resource manager builds on DS mechanisms to support two premium flow types within a single framework: latency- and jitter sensitive (e.g., media flows) and high-bandwidth but latency-insensitive (e.g., bulk transfer). We also propose a policy model that allows admission control decisions to be made at multiple levels. Finally, we present performance experiments conducted on both local area and wide area DS network testbeds; these demonstrate our ability to support multiple flow types and to co-reserve network and CPU resources.

The rest of this article is structured as follows. In Section 2 we use three high-end applications to derive requirements for QoS mechanisms. Then, in Section 3, we describe GARA and its implementation. In Section 4, we describe its application in the context of DS networks and in Section 5 we present our experimental results. We discuss multi-domain issues in Section 6 and related work in Section 7, and conclude with a discussion of future directions.

2 QoS Requirements of High-End Applications

We describe three representative examples of the high-end network applications that are encountered, for example, in advanced scientific and engineering computing [7], and then list what we see as their key QoS requirements.

2.1 Application Descriptions

Distance visualization of large datasets. Scientific instruments and supercomputer simulations generate large amounts of data: tens of terabytes today, petabytes within a few years. Remote interactive exploration of such datasets requires that the conventional visualization pipeline be decomposed across multiple resources. A realistic configuration might involve moving data at hundreds or thousands of Mb/s to a data analysis and rendering engine which then generates and streams real-time MPEG-2 (or later HDTV) video to remote client(s), with control information flowing in the other direction. QoS parameters of particular interest for this class of application include bandwidth, latency and jitter; resources involved in delivering this QoS include storage, network, CPU, and visualization engines.

Large data transfers. In other settings, large datasets are not visualized remotely but instead are transferred in part or in their entirety to remote sites for storage and/or analysis. The need to coordinate the use of other resources with the completion of these multi-gigabyte or terabyte transfers leads to a need for QoS guarantees of the form "data delivered by deadline" rather than instantaneous bandwidth. Notice that achieving this goal requires the scheduling of storage systems and CPUs as well as networks so as to achieve often extremely high transfer rates.

High-end collaborative environments. High-end collaborative work environments involve immersive virtual reality systems, high-resolution displays, connections among many sites, and multiple interaction modalities including audio, video, floor control, tracking, and data exchange. For example, the NCSA Alliance "Access Grid" currently connects some 15 sites via multiple audio, video, and control streams, with the audio streams especially vulnerable to loss. Such applications require QoS mechanisms that allow the distinct characteristics of these different flows to be represented and managed [3].

2.2 QoS Requirements

Heterogeneous flows. The applications of interest frequently incorporate multiple flows with widely varying characteristics, in terms of bandwidth, latency, jitter, reliability, and other requirements. GARA addresses these requirements through (a) support for per-flow QoS specifications while maintaining DS-like scalability and (b) a QoS-mechanism-independent architecture that adapts to multiple techniques. A common API means that for example a distance visualization application can specify the distinct requirements of high-volume data and latency-sensitive control flows, in a mechanism-independent manner; these flows might then be mapped to different mechanisms: e.g., MPLS and DS.

High bandwidth flows. Some applications involve high bandwidth flows that may require a large percentage of the available bandwidth on a high-speed link. For example, in recent work, Rebecca Nitzan and Brian Tierney of Lawrence Berkeley National Laboratory (LBNL) demonstrated transfer rates of 450 Mb/s over a wide area OC12 network. This characteristic has significant implications for both mechanisms and policy. QoS mechanisms are required that can support such flows while allowing coexistence with other flows having different characteristics. At the policy level, we believe that approaches are required that allow for the coordinated management of resources in multiple domains, so that virtual organizations (e.g., a scientific collaboration) can express policies that coordinate the allocation of the resources available to them in different domains.

Need for end-to-end QoS. Satisfying application-level QoS requirements often requires the coordinated management of reousrces other than networks: for example, a high-speed data transfer can require the scheduling of storage system, network, and CPU resources. As we shall see, GARA addresses this requirement by defining an extensible architecture that can deal with a range of different resource types and by providing support for the co-allocation of multiple resources.

Need for application-level control. High end-to-end performance requires that applications be able to discover resource availability (GARA), monitor achieved service, and modify QoS requests (to network and to other resources, such as CPUs) and application behavior dynamically.

Need for advance reservation. Specialized resources required by high-end applications such as high-bandwidth virtual channels, scientific instruments and supercomputers are in scarce and in high demand; in the absence of advance reservation mechanisms, coordination of the necessary resources is difficult. Reservation mechanisms are needed to ensure that resources and services may be scheduled in advance.

3 The GARA QoS Architecture

We designed the Globus Architecture for Reservation and Allocation (GARA) to meet the QoS requirements listed above. Here we introduce GARA concepts; we describe below how we apply these concepts in DS networks to manage the allocation of a particular flavor of QoS capability, namely premium service as described in RFC2598.

3.1 GARA Overview

GARA defines APIs that allows users and applications to manipulate reservations of different resources in uniform ways. For example, essentially the same calls are used to make an immediate or advance reservation of a network or CPU resource. Once a reservation is made, an opaque object called a reservation handle is returned that allows the calling program to modify, cancel, and monitor the reservation. Other functions allow reservations to be monitored by polling or through a callback mechanism in which a user's function is called every time the state of the reservation changes in an interesting way.

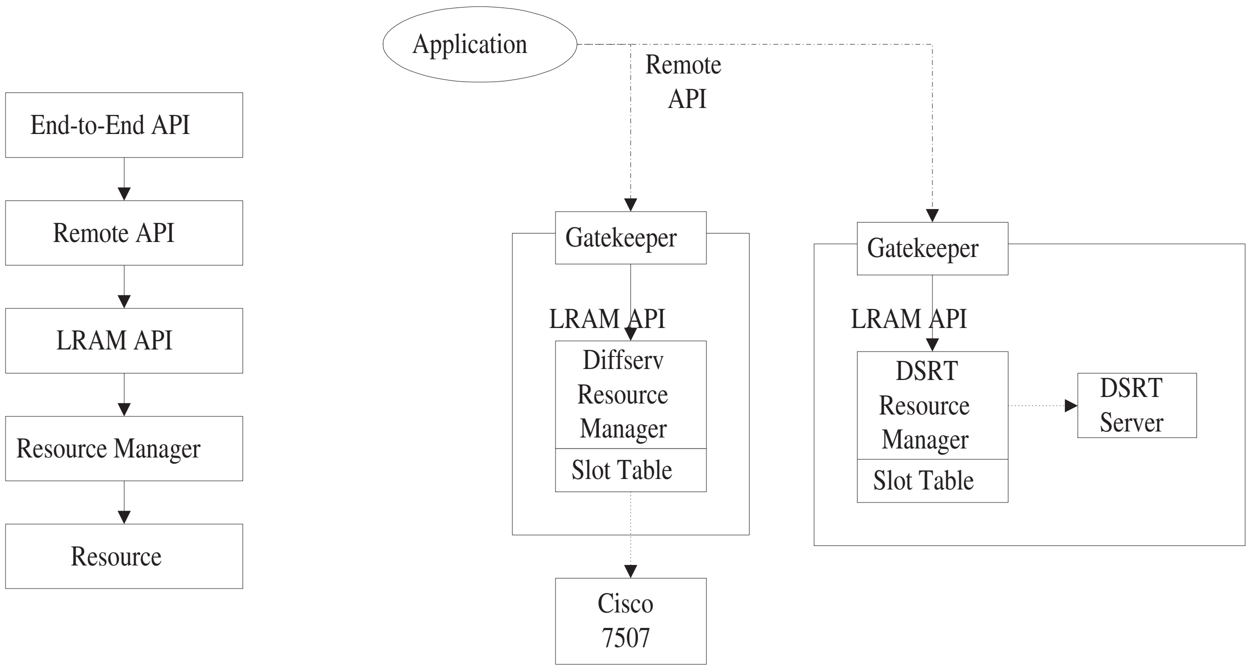

Figure 1: On the left, the principal APIs used within GARA. On the right, the principal components of the GARA prototype as instantiated for DS and DSRT (CPU scheduler) services, with our own resource manager and slot manager being used in both cases. In the DS case, commands are issued to a router while in the DSRT case commands are issued to a DSRT server.

As illustrated on the left side of Figure 1, GARA defines APIs at multiple levels so as to maximize both the functionality delivered to the user and opportunities for code reuse in implementations. In particular, the Local Reservation and Allocation Manager (LRAM) API provides direct access to reservation functions within a trust domain, while the remote API provides remote access to LRAM functionality, addressing issues of authentication and authorization. Both APIs implement the functionality described in the preceding paragraph.

The uniform treatment of reservations provided by GARA makes it possible to define and reuse co-reservation and co-allocation libraries that encode strategies for the coordinated use of multiple resources [2]. Because different resources (e.g., computers and storage systems) can be manipulated via the same function calls, standard libraries can be developed that encode (for example) fault recovery strategies.

One co-reservation library that we have developed in support of our work with GARA implements an end-to-end network API that provides end-to-end analogs of each of the remote API calls. This API allows the user to create, monitor, cancel, etc., network co-reservations: that is, reservations involving more than one network resource. This API allows users and applications to ignore details of the underlying network topology.

```
UDP-streamer(host A, host B) {
   (PortA,PortB) = establish_socket_connection(A, B)
   F = compute_flow_requirement()
   Max = EnquireE2EResv(A, B, {NOW,60 mins})
   if (Max.forward > F) then R = F else R = Max.forward
   H = CreateE2EResv(A, B, R, 0, {NOW,60 mins})
   BindE2EResv(H, PortA, PortB)
   repeat until done {
      <send for a while>
      Max = EnquireE2EResv(A, B, {NOW,60 mins})
      if (Max.forward > 0 && R < F) then {
         R = Max.forward + R
         if (R > F) then R = F
         ModifyE2EResv(H, R, NOW, 60 mins)
      }
   }
}
```

Figure 2: Pseudo-code for a simple application that uses the GARA end-to-end API to first make and subsequently monitor and modify a reservation. For brevity, this code does not include error checking.

Figure 2 illustrates the use of this end-to-end API. This program first determines the bandwidth requirements of an application and then queries to determine available premium bandwidth over the path of interest. A reservation is created for the smaller of these two values and the reservation handle `H` is used to bind the reservation to the previously created flow. The application then checks periodically to see whether the reservation can be increased. Notice that the changes to what is otherwise a conventional socket-based code are small.

We note that while this example emphasizes application-centered monitoring and control of reservation state, GARA also supports third-party reservation operations. For example, we could remove the reservation logic from Figure 2 altogether and instead perform appropriate reservation operations in a separate process.

3.2 GARA Implementation

We review GARA implementation issues and status, working from the bottom of our API stack.

GARA must provide admission control and reservation enforcement for multiple resources of different types. Because few resources provide reservation capabilities, we have implemented our own resource manager so as to ensure availability of reservation functions. As illustrated in Figure 1, this manager uses a slot table [4, 11] to keep track of reservations and invokes resource-specific operations to enforce reservations. Requests to this resource manager are made via the LRAM API and result in calls to functions that add, modify, or delete slot table entries; timer-based callbacks generate call-outs to resource-specific routines to enable and cancel reservations. Note that only certain elements of this resource manager need to be replaced to instantiate a new resource interface. To date, we have developed resource managers for DS networks (described below), for the Distributed Soft Real-Time (DSRT) CPU scheduler [12], and for the Distributed Parallel Storage System (DPSS), a network storage system; others are under development.

Our implementation of the end-to-end API invokes a path service to identify the resource managers that must be contacted to arrange for an end-to-end reservation, and then makes a series of GARA remote API calls to perform the co-reservation oepration. See below for a discussion of issues that arise when traversing multiple domains.

Our GARA prototype uses two "Grid" services provided by the Globus toolkit: the Globus Lightweight Directory Access Protocol (LDAP)-based information service for publishing reservation status information and for accessing path information, and the public-key based Grid Security Infrastructure for authentication and authorization services. The interfaces to these services are simple and well-defined (LDAP and GSS-API, respectively), hence it is straightforward to substitute alternative implementations.

4 GARA and Differentiated Services Networks

The DS architecture is based on a simple model where packets entering a network are classified and possibly conditioned at the boundaries of the network by service provisioning policies, and assigned to different behavior aggregates. Within the core of the network, packets are forwarded according to the per-hop behavior (PHB) associated with the DS classification. These mechanisms have the advantage of not requiring that per-flow state be maintained within the network. However, few guarantees can be made about end-to-end behaviors, which emerge as the composition of the PHBs associated with individual links.

4.1 Integrating Differentiated Services and GARA

We have interfaced GARA concepts and constructs to DS mechanisms in order to manage the allocation of premium service bandwidth. As shown in Figure 3, we associate GARA resource managers with the locations at network edges where admission control occurs. These resource managers are, in essence, what DS papers call "bandwidth brokers": they allocate their region's marked (premium) traffic allocations and control the devices (e.g., routers) used to enforce these allocations. Requests to resource managers are authenticated, ensuring secure operation.

We have constructed our DS resource manager to support two classes of premium service: a foreground service, for latency- and jitter-sensitive flows (e.g., multimedia streaming and control), and a background service, for long-lived, high bandwidth but latency-insensitive flows (e.g., bulk data transfer operations). The resource manager changes background reservations dynamically as foreground reservations come and go, generating callbacks to the application when a reservation changes. This strategy allows bulk data transfers to co-exist with multimedia flows. The amount of bandwidth available for background reservations over a particular time period can then be controlled via policy mechanisms. We report results with this approach below. Our prototype supports multiple foreground reservations but initially only a single background reservation; the extensions required to support multiple background flows are not complex.

A resource management framework for DS networks must also address end-to-end issues. A typical wide area flow requires allocations of premium bandwidth at multiple edge routers and also within interior domains. For example, in Figure 3, a premium flow from ANL to LBNL should, in principle, require an allocation not only from the ANL domain for the ANL/ESnet interface (where marking occurs) but also from ESnet for the ANL-LBNL transit traffic and from the LBNL domain for the ESnet/LBNL interface. Hence, we need to associate resource managers with multiple DS domains and to implement co-reservation strategies. Co-reservation operations must be designed with end-to-end verification in mind. In our example, an application that omitted to obtain a reservation for ESnet transit traffic could cause problems for other ANL-LBNL traffic, for example if the aggregate ANL-ESnet traffic exceeded what was allowed by the current ANL-ESnet service level agreement (SLA).

Most DS work assumes that co-reservation operations are encapsulated in the local domain's resource manager: hence, a request to reserve bandwidth from ANL to LBNL results in the ANL manager contacting the ESnet manager, which in turn contacts the LBNL manager. Upon receipt of a positive response from both other managers, a reservation is established. This approach has the advantages of providing trusted co-reservation and of encapsulating all bandwidth broker communication within a single local entity. The approach has disadvantages in settings where end-to-end reservations involve resources other than networks, as a hierarchical co-reservation structure results, or where allocation policies at interior domains depend on factors other than the identity of the requesting manager.

An alternative approach to this problem is to define a two-phase commit protocol. In this approach, an application program—or agent working on behalf of an application program—contacts each manager in turn. In the first phase, a manager can indicate that acceptance of a reservation is conditional on the requestor securing acceptance (indicated by a signed certificate) from the next manager.

In both approaches, interdomain SLAs can either be established statically (in which case reservations can only be made if they fit within the pre-established SLAs), or they can be established dynamically, as reservations are made. The latter approach provides greater flexibility but requires more sophisticated policy and enforcement engines in interior domains, as discussed below.

Our initial GARA prototype implements neither of the approaches just described but instead relies on the end-to-end library to implement co-reservations correctly. We assume two domains and static SLAs between domains; hence, we need to allocate bandwidth at just two locations. Reservation policies are expressed via access control lists associated with individual resource managers. These limitations are not inherent in our model and are being removed in current work.

4.2 Differentiated Service Configuration

The final issue to be addressed in a DS implementation relates to how PHBs are configured to provide the premium services desired for particular applications. In our DS implementation, this configuration involves the Committed Access Rate (CAR) and Weighted Fair Queueing

(WFQ) mechanisms supported by Cisco Systems 7500 Series routers.

We use CAR on the ingress ports of edge routers to mark and police the flows for which premium bandwidth is required. Specifically, we use CAR to first rate limit (dropping non-conforming packets) and then mark packets for specified flows. As CAR currently does not support the DS Codepoints defined in RFC2474, we use the IP Precedence field to mark packets belonging to premium service flows.

The operation of CAR is controlled via commands issued to the router by the associated GARA resource manager as reservations become active, terminate, are modified, or are cancelled. These commands enter, remove, or modify flow specifications that define a premium service flow in terms of its source and destination IP address and port, and its rate limit specification (desired average transmission rate bandwidth and a normal and excess burst size). Communication from the resource manager to the Cisco Systems router is performed via Command Line Interface.

We also use CAR on the ingress ports of inter-domain routers, where it is used to enforce SLAs negotiated with other domains, by rate limiting the precedence-marked traffic that will be accepted from another domain.

We use WFQ on the egress port of edge routers and in interior routers. WFQ ensures that in periods of congestion—i.e., when packets get queued in the router because the output link does not provide the capacity for delivering them immediately—each IP precedence class receives at least the fraction of the output bandwidth defined by the weight defined for that class. Hence, as long as the total marked traffic destined for an output port does not exceed the allocated output bandwidth, WFQ can be used to ensure that marked traffic is forwarded without delay despite congestion.

This use of CAR and WFQ approximates the Expedited Forwarding (EF) PHB described by the IETF's DS Working Group in RFC2598, but does not match it perfectly. In particular, packets marked with the EF PHB should always be treated before other packets, but we merely guarantee a portion of the bandwidth, not a priority service. We considered using the Priority Queueing mechanism available in Cisco Systems routers, but it does not work at the higher speeds required by our applications.

This use of CAR and WFQ raises the question of how these mechanisms should be configured to meet application-level QoS requirements. This question is complicated by the wide variety of flows that we wish to support (UDP, TCP, low and high bandwidth) and the geographic scale over which QoS is required: from a few meters to thousands of kilometers. Considerable experimentation on the testbed described in the next section has been performed to understand these issues.

5 Experimental Studies

We report on experiments designed to evaluate the effectiveness of both the GARA architecture and our DS implementation.

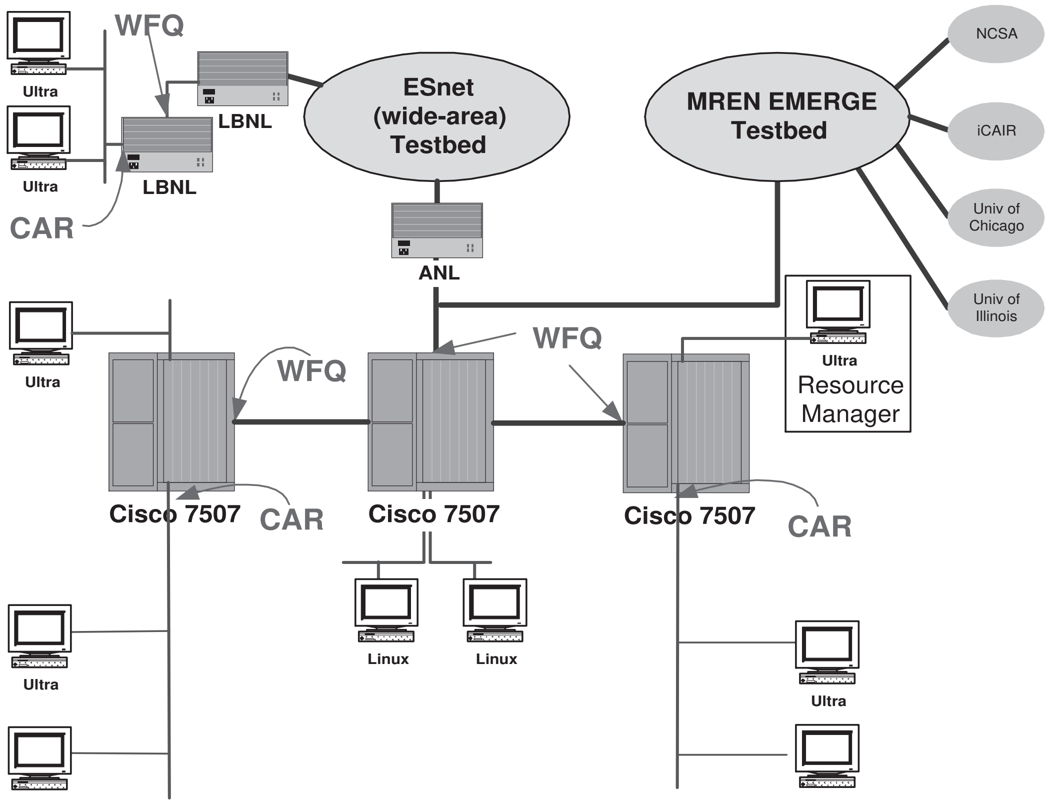

Figure 3: The experimental configuration used in this work, showing our local GARNET testbed and its extensions to remote sites connected via ESnet and MREN.

5.1 Experimental Configuration

Our experimental configuration, illustrated in Figure 3, comprises a laboratory testbed at Argonne National Laboratory (the Globus Advance Reservation Network Testbed: GARNET) connected to a number of remote sites, including Lawrence Berkeley National Laboratory (LBNL). Connectivity to LBNL is provided by the Energy Sciences Network (ESnet) DS testbed. GARNET allows controlled experimentation with basic DS mechanisms; the wide area extensions allow for more realistic operation, albeit with a small number of sites. Notice that end-system resources are located in different domains; hence, we must deal with distributed authentication and authorization.

Cisco Systems 7507 routers are used for all experiments. Within GARNET, these are connected by OC3 ATM connections; across wide area links, they are connected by VCs of varying capacity. We are restricted to these relatively slow speeds because the 7507 cards do not implement CAR and WFQ at speeds faster than OC3. End system computers are connected to routers by either switched Fast Ethernet or OC3 connections. CAR and WFQ are used for QoS enforcement, as described above. Flow specifications supplied to CAR use a bandwidth computed from the user-specified required bandwidth, taking into account packet headers (note that this requires packet size information), with nonconforming traffic dropped. Burst size and excess burst size parameters are both set as follows: if using TCP, to the bandwidth (in bytes/second) times the assumed maximum round trip time, subject to a minimum of 8 Kbytes and a maximum of 2 Mbytes; if using UDP, to 1/4 of this value, subject to a minimum of 8 Kbytes. WFQ was configured statically in all experiments.

No traffic shaping is performed on premium flows beyond the limited shaping provided by WFQ in the presence of congestion. While the lack of shaping has not proven to be a significant problem to date, it will likely be required in future, more dynamic environments.

The network speeds supported in this testbed are clearly not adequate for the high-end applications discussed above: the largest premium flow that we can support is around 80 Mb/s. Nevertheless, this testbed configuration has allowed us to validate multiple aspects of our general approach. We plan to extend our approach to higher-speed networks in future work.

5.2 Multiple Flows: Local Area Case

Our first experiments evaluate our ability to support multiple flows simultaneously and to support application monitoring of, and adaptation to, changes in reservation status.

We first report on experiments conducted on our local GARNET testbed: see Figure 3. We configured GARNET to create a 45 Mb/s premium channel in a 100 Mb/s network. We then created five distinct flows: a bulk data transfer, operating as a "background" flow; a competing 80 Mb/s best-effort UDP flow (a traffic generator submitting 1,000 byte packets every 100 μsecs); and three independent, short-lived foreground flows with immediate reservations. In this and subsequent experiments we used a simple data transfer program, ttcp, as our "application." (Experiments with more complex applications have started and give promising results; these will be reported in subsequent work.) The premium and competing flows are sourced and sinked by different computers.

Figure 4 shows the bandwidth delivered to the foreground, background, and best effort flows during the experiment. We succeed in delivering "excess" premium bandwidth to the bulk transfer application without comprising the foreground flows. The good bulk transfer performance achieved is made possible by the manager's callbacks to the bulk transfer application, which allow that application to change its sending rate in response to changes in its allocating bandwidth, hence avoiding packet drops and invocation of TCP slow-start. The following is a more detailed explanation of the graph:

1. The graph begins with the background TCP traffic, which has a bulk-transfer reservation. This flow is initially allocated 40.5 Mb/s premium bandwidth: that is, 90 percent of the 45 Mb/s premium traffic.

2. The competitive UDP traffic is started shortly after the bulk transfer but does not affect it due to the premium status of the bulk transfer flow.

3. At 25 secs, another application makes an immediate 36 Mb/s reservation and initiates a 32 Mb/s foreground flow. A callback notifies the bulk transfer application, which reduces its sending rate to adapt to the reduced reservation. (The and other similar transitions take a little time due to the time required to control the router.)

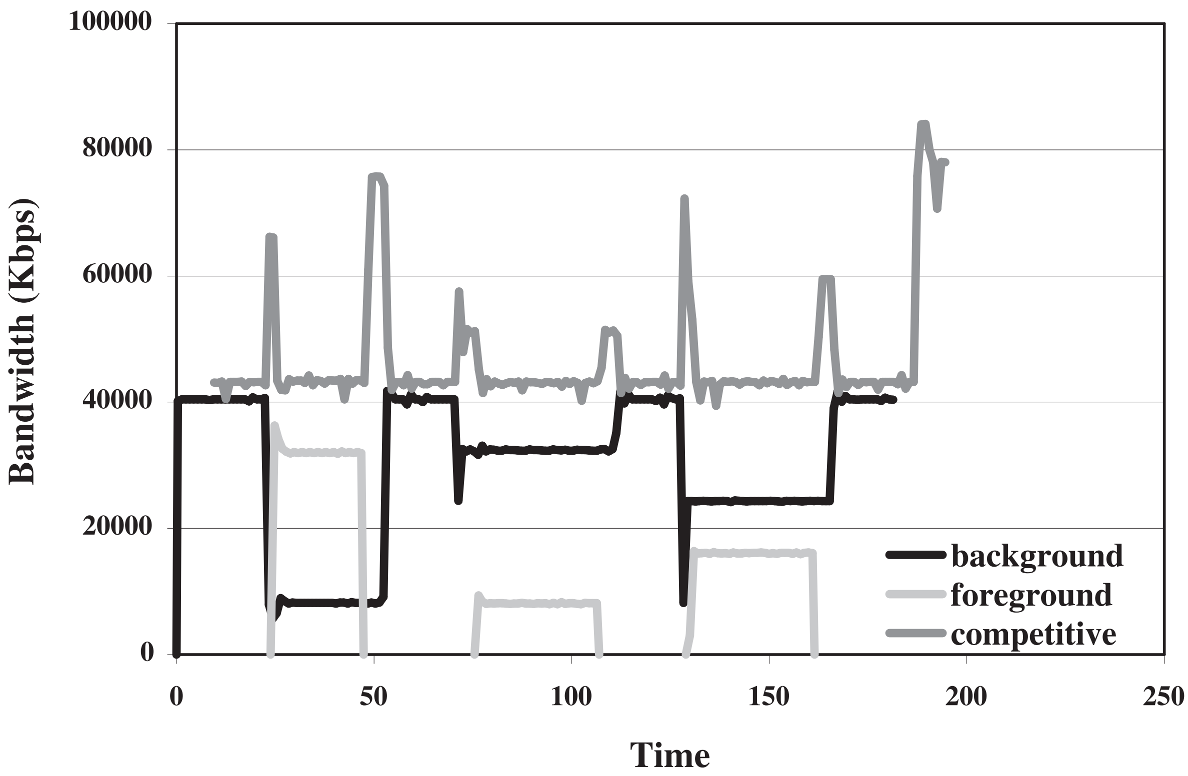

Figure 4: Performance achieved for a mixture of premium and best effort services on GARNET. We demonstrate that a bulk-transfer (background) application is able to exploit unused premium traffic without affecting foreground reservations. See text for details.

4. At 48 secs, the foreground application finishes its transmission and then cancels its reservation. Another callback allows the bulk transfer process to increase its sending rate to adapt to the newly available premium traffic.

5. Subsequently, two other foreground flows are created, with similar effects: a 9 Mb/s reservation (8 Mb/s flow) from 75 to 105 secs and an 18 Mb/s reservation (16 Mb/s flow) from 130 to 160 secs.

6. At time 185, the background flow completes and cancels its reservation. The competing traffic rate increases to its target of 80 Mb/s, actually exceeding this briefly because of the filled router queues.

Notice that each time the bulk transfer reservation is reduced, the bulk-transfer rate drops momentarily then recovers. We attribute this behavior to the fact that TCP shrinks its window size when packets are dropped, either by falling into its slow start phase or into its congestion avoidance phase.

5.3 Multiple Flows: Wide Area Case

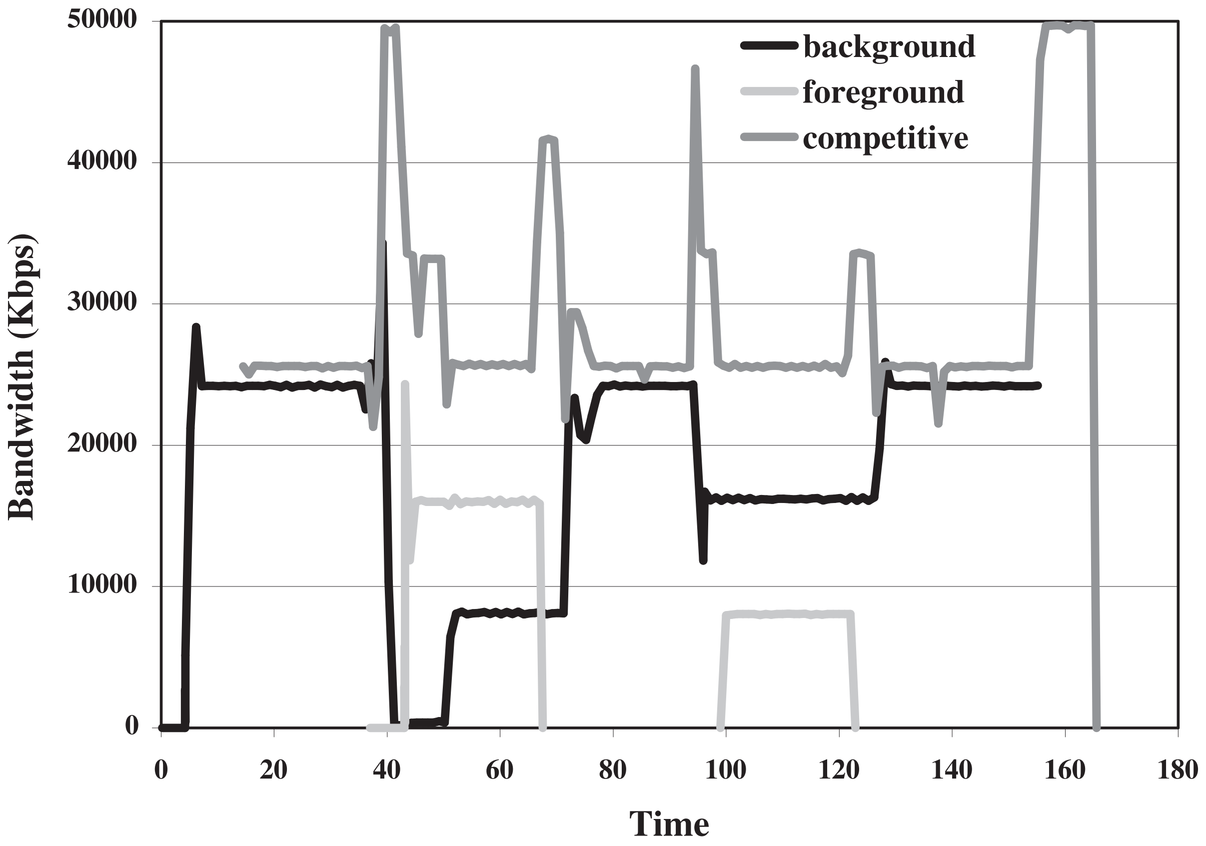

Figure 5: Performance achieved for a mixture of premium and best effort services on a wide area testbed. We demonstrate good performance even in the wide area. See text for details.

Our next experiments repeat those just described over the wide area network from ANL to LBNL: see Figure 3. Here, we used WFQ to configure our testbed with 55 Mb/s premium traffic over the 60 Mb/s UBR VC between ANL and LBNL and 27 Mb/s premium traffic within GARNET. Note that when congested this wide area premium traffic configuration is a good approximation to priority queuing. (Only 27 Mb/s premium traffic was allowed on GARNET in these particular experiments because of either extra traffic or a bad device on a fast Ethernet segment of the network that we were unable to control; in other experiments, we have successfully configured up to 45 Mb/s premium.) Here, the background flow is initially allocated 24.3 Mb/s premium bandwidth (that is, 90 percent of 27 Mb/s), the competing best-effort UDP flow operates at 50 Mb/s (1,250 byte packets every 200 μsecs), and two foreground flows are created: a 16 Mb/s flow (18 Mb/s reservation) at 37 secs and an 8 Mb/s flow (9 Mb/s reservation) at 94 secs.

As shown in Figure 5, the results obtained in the wide area are almost as good as in the local area. We attribute the somewhat more dynamic behavior during reservation changes to the fact that the kernel buffers associated with the bulk transfer socket take some time to empty. Hence, data is initially sent too rapidly for the updated router configuration, forcing packets to be dropped and TCP to go into slow-start mode. This effect is magnified by the larger bandwidth-delay product and hence larger socket buffers (1 MB in this case) in the wide area network.

5.4 Co-Reservation of CPU and Network

An important challenge addressed by GARA is the co-reservation of multiple resources: for example, network and CPU to ensure that a receiver can process incoming data. The experiment reported here demonstrates our ability to support such co-reservation. Specifically, we establish a TCP flow and show that we can maintain data transfer performance despite competing traffic on the network and competing computational load on the receiving host.

We conducted this experiment on GARNET and use the 100 Mb/s network as before, except that this time premium traffic is configured to use up to 95 Mb/s. A TCP flow is started and network and CPU reservations and load are applied in various combinations.

1. A ttcp application is started without network congestion and without any reservation.

2. At 10 secs, an 80 Mb/s traffic generator is started. Because of network congestion, ttcp switches to the slow start feature and congestion control, with the result that ttcp performance drops precipitiously and most available bandwidth is consumed by the competitive traffic.

3. At 40 secs, the TCP application creates an immediate network reservation through GARA. Performance increases dramatically.

4. At 60 secs, a significant competing CPU load is imposed on the TCP receiver host. TCP throughput is significantly effected, due to contention for the CPU.

5. At 80 secs, we use GARA to reserve a significant amount of CPU for the receiving TCP process through the DSRT manager. The achieved rate increases immediately, although some variation remains due to the interval-based scheduling used by DSRT.

6. At 120 secs, we cancel the network reservation; TCP performance drops precipitiously once again.

7. At 160 secs, we cancel the CPU reservation; this has little further impact on performance.

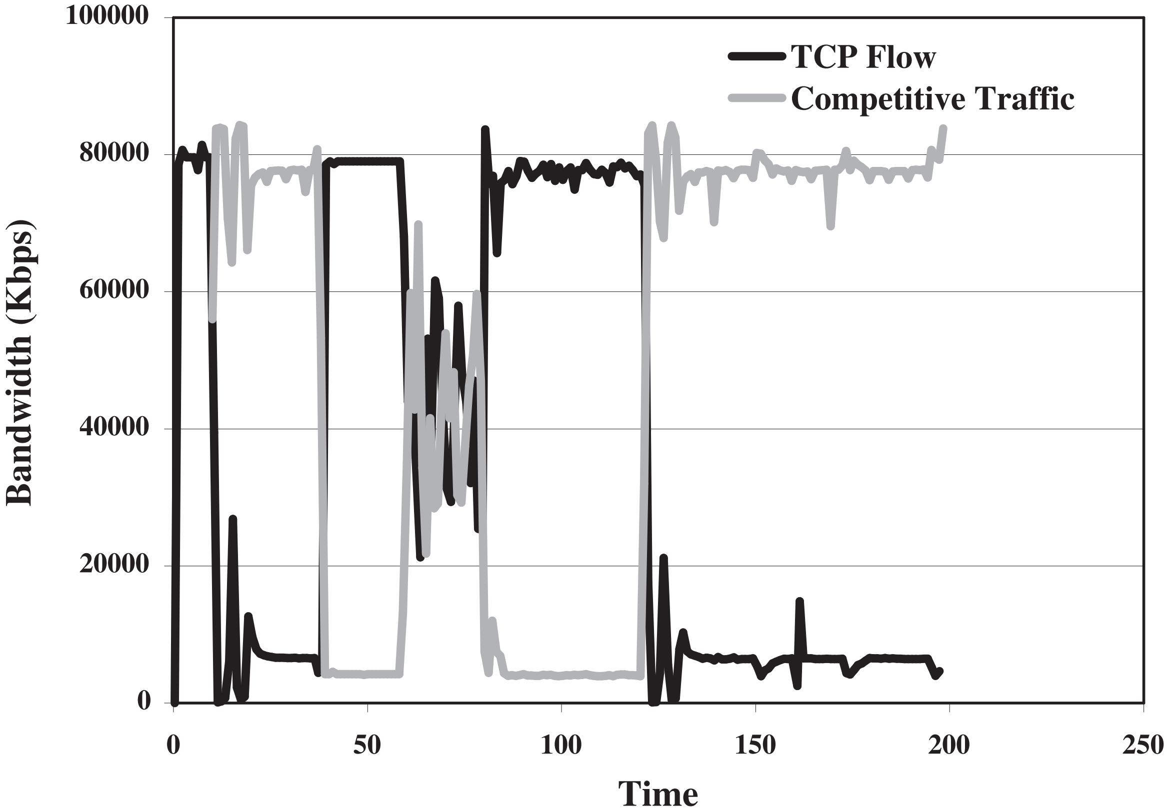

Figure 6: Performance achieved for a TCP flow in the presence of competing UDP traffic and host load, for various combinations of network and CPU reservation. We demonstrate GARA's ability to co-reserve multiple resource types.

6 Policy in Multidomain Settings

We sketch here the approach that we propose to take in the future when expanding GARA to support more sophisticated policy enforcement, particularly in multi-domain settings.

We assume the following system model. An end-to-end reservation may involve multiple resources located in different domains. Resource allocation decisions within a domain remain the responsibility of that domain; hence, end-to-end reservations must be authorized by all appropriate domains or by entities to which domains have delegated this authority.

Our policy approach is designed to support a flexible mix of policy options, for example:

- A domain may allocate resources on the basis of user identity. Such a policy may be appropriate in the case of unique resources for which users make distinct requests, e.g., supercomputers or specialized network resources such as a low-bandwidth outgoing connection.

- A domain may allocate resources in response to a request forwarded from another domain with which some agreement has been negotiated previously. For example, an transit service domain (e.g., ESnet in Figure 3) might negotiate an agreement to accept any allocation request forwarded from another DS domain, up to some SLA limit.

- A domain may allocate resources in response to a request authorized by some third party, such as a virtual organization with which the domain has negotiated an agreement previously [1]. This delegation of authorization allows a community to negotiate agreements with multiple domains in order to obtain control of some amounf of premium end-to-end bandwidth.

We anticipate multiple such authorization policies being active at one time. For example, in an environment such as that of Figure 3, a transit domain such as ESnet might support the following policies:

- Accept immediate reservations of premium bandwidth from any domain with a previously negotiated SLA, subject to the constraint that no single request can be more than 100 Mb/s and the total requests from a domain cannot exceed its SLA.

- Accept immediate and advance reservation requests labeled as "HEP" if approved by a server operated by the high energy physics community, up to limits and at times previously negotiated with that community.

We believe that authorization and authentication mechanisms provided in the Globus toolkit provide a basis on which to explore these issues. The Akenti system [17] also provides important relevant technology.

7 Related Work

The general problem of QoS implementation and management is receiving increased attention (see, e.g., [9]). However, there has been little work on the specific problems addressed in this paper, namely advance reservation and co-reservation of heterogeneous collections of resources for end-to-end QoS and the use of DS mechanisms to support flow types encountered in high-end applications.

Proposals for advance reservations typically employ cooperating servers that coordinate advance reservations along an end-to-end path [18, 5, 4, 10]. Techniques have been proposed for representing advance reservations, for balancing immediate and advance reservations [5], for advance reservation of predictive flows [4]. However, this work has not addressed the co-reservation of resources of different types.

The concept of a bandwidth broker is due to Jacobson. The Internet 2 Qbone initiative and the related Bandwidth Broker Working Group are developing testbeds and requirements specifications and design approaches for bandwidth brokering approaches intended to scale to the Internet [16]. However, advance reservations do not form part of their design. Other groups have investigated the use of DS mechanisms (e.g., [19]) but not for multiple flow types.

The co-reservation of multiple resource types has been investigated in the multimedia community: see, for example, [13, 15, 14]. However, these techniques are specialized to specific resource types.

8 Conclusions and Future Work

We have described a QoS architecture that supports immediate and advance reservation (and co-reservation) of multiple resource types; application-level monitoring and control of QoS behavior; and support for multiple concurrent flows with different characteristics. We have also described how this architecture can be realized in the context of differentiated service networks. We presented experimental results that demonstrate our ability to deliver QoS to multiple flows in local and wide area networks.

In future work we plan to improve and extend GARA in a variety of areas, including improved representation and implementation of policy, more sophisticated adaptation mechanisms (including real-time monitoring of network status), and more sophisticated co-reservation algorithms [2]. We also plan to extend our evaluation of GARA mechanisms to a wider range of applications and more complex networks.

Acknowledgments

We gratefully acknowledge assistance provided by Rebecca Nitzan and Robert Olson with experimental studies. Numerous discussions with our colleagues Gary Hoo, Bill Johnston, Carl Kesselman, and Steven Tuecke have helped shape our approach to quality of service. We also thank Cisco Systems for an equipment donation that allowed creation of the GARNET testbed. This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation; and by the NASA Information Power Grid program.

References

[1] S. Blake, D. Black, M. Carlson, M. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *Internet RFC 2475*, 1998.

[2] Karl Czajkowski, Ian Foster, and Carl Kesselman. Co-allocation services for computational grids. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*. IEEE Computer Society Press, 1999.

[3] Tom DeFanti and Rick Stevens. Teleimmersion. In *[7]*, pages 131–156.

[4] M. Degermark, T. Kohler, S. Pink, and O. Schelen. Advance reservations for predictive service in the internet. *ACM/Springer Verlag Journal on Multimedia Systems*, 5(3), 1997.

[5] D. Ferrari, A. Gupta, and G. Ventre. Distributed advance reservation of real-time connections. *ACM/Springer Verlag Journal on Multimedia Systems*, 5(3), 1997.

[6] I. Foster and C. Kesselman. Globus: A toolkit-based grid architecture. In *[7]*, pages 259–278.

[7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[8] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, pages 27–36, 1999.

[9] Roch Guérin and Henning Schulzrinne. Network quality of service. In *[7]*, pages 479–503.

[10] A. Hafid, G. Bochmann, and R. Dssouli. A quality of service negotiation approach with future reservations (nafur): A detailed study. *Computer Networks and ISDN Systems*, 30(8), 1998.

[11] G. Hoo, W. Johnston, I. Foster, and A. Roy. QoS as middleware: Bandwidth broker system design. Technical report, LBNL, 1999.

[12] Hao hua Chu and Klara Nahrstedt. CPU service classes for multimedia applications. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*. IEEE Computer Society Press, 1999.

[13] A. Mehra, A. Indiresan, and K. Shin. Structuring communication software for quality-of-service guarantees. In *Proc. of 17th Real-Time Systems Symposium*, December 1996.

[14] K. Nahrstedt, H. Chu, and S. Narayan. QoS-aware resource management for distributed multimedia applications. *Journal on High-Speed Networking, IOS Press*, December 1998.

[15] K. Nahrstedt and J. M. Smith. Design, implementation and experiences of the OMEGA end-point architecture. *IEEE JSAC, Special Issue on Distributed Multimedia Systems and Technology*, 14(7):1263–1279, September 1996.

[16] B. Teitelbaum, S. Hares, L. Dunn, V. Narayan, R. Neilson, and F. Reichmeyer. Internet2 QBone - Building a testbed for differentiated services. *IEEE Network*, 13(5), 1999.

[17] Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, and Abdelilah Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the Eighth Usenix Security Symposium*. 1999.

[18] L.C. Wolf and R. Steinmetz. Concepts for reservation in advance. *Kluwer Journal on Multimedia Tools and Applications*, 4(3), May 1997.

[19] Ikjun Yeom and A. L. Narasimha Reddy. Modeling tcp behavior in a differentiated-services network. Technical report, TAMU ECE, 1999.