PROPOSAL

Section I.

Reference: Office of Science, Notice 99-08

Technical Topic Areas:
High-speed networks, high-performance and real-time I/O subsystems and QoS-enabled middleware

QoS-enabled Middleware for High-Speed Networks and Endsystems

Prepared for:

U.S. Department of Energy
Office of Science
Grants and Contracts Division,
SC-64, 19901 Germantown Road,
Germantown, MD 20874-1290, Attention: Program Notice 99-08

Prepared by:

Douglas C. Schmidt, Associate Professor of Computer Science
Jonathan S. Turner, Professor of Computer Science
Fred Kuhns, Senior Research Associate of Computer Science
David Levine, Senior Research Associate of Computer Science

Points of Contact:

Technical Matters:
Douglas C. Schmidt, Director, Center for Distributed Object Computing
Dept of Computer Science, Washington University in St. Louis
(314) 935-7538, FAX (314) 935-7302, email: schmidt@schmidt.wustl.edu

Administrative Matters:
Jonathan S. Turner, Director, Applied Research Laboratory
Dept of Computer Science, Washington University in St. Louis
(314) 935-6132, FAX (314) 935-7302, email: jst@cs.wustl.edu

Proposal Abstract

Our proposed effort is aimed at the design, prototype implementation, and demonstration of an integrated high-speed networking and middleware infrastructure. This infrastructure will provides QoS specification and enforcement features to *program*, *provision*, and *control* advanced, high-speed networks, and QoS-enabled middleware for Next Generation Internet (NGI) applications. Our architecture (shown in Figure 1) integrates and enhances the following technologies we have developed under previous and ongoing sponsorship:

A very high-speed network infrastructure operating at Gigabit speeds: This infrastructure includes (1) the high-performance WUGS switch with hardware support to efficiently handle multicast traffic, (2) an advanced ATM interface capable of link speed in excess of 1 Gbps and (4) intelligent port controllers for the switch that provide network layer processing, as well as support for QoS mechanisms and active networks.

Horizontally and vertically integrated dynamic and adaptive QoS guarantees: Our infrastructure will provide QoS guarantees throughout various components and domains on an endsystem. At the user-level, an OO communication framework provides high-performance, QoS-enabled middleware that provides a uniform interface for provisioning, monitoring, and controlling the WUGS high-speed network elements. In addition, NGI applications can use this middleware to reserve CPU, memory, and I/O resources.

1 Introduction

1.1 Motivation and Research Objectives

During the past decade, there has been substantial R&D emphasis on *high-speed networking* and *performance optimizations* for network elements and protocols. This effort has paid off such that networking products are now available off-the-shelf that can support Gbps on every port, *e.g.*, Gigabit Ethernet and ATM switches. Moreover, 622 Mbps ATM connectivity in WAN backbones are becoming standard and 2.4Gbps is starting to appear. In networks and GigaPoPs being deployed for the Next Generation Internet (NGI), such as the Advanced Technology Demonstration Network (ATDnet) [1], 2.4 Gbps (OC-48) link speeds are being deployed. However, the general lack of robust and flexible tools and middleware for programming, provisioning, and controlling these networks has limited the rate at which NGI applications have been developed to leverage advances in high-speed networks.

During the same time period, there has also been substantial R&D emphasis on object-oriented (OO) communication *middleware*, including open standards like OMG's Common Object Request Broker Architecture (CORBA) [2], as well as popular proprietary solutions like Microsoft's Distributed Component Object Model (DCOM) [3] and Sun's Remote Method Invocation (RMI) [4]. These efforts have paid off such that OO middleware is now available off-the-shelf that allows clients to invoke operations on distributed components without concern for component location, programming language, OS platform, communication protocols and interconnects, or hardware [5]. However, the general lack of support in off-the-shelf middleware for QoS specification and enforcement features; integration with high-speed networking technology; and performance, predictability, and scalability optimizations [6] has limited the rate at which NGI applications have been developed to leverage advances in OO middleware.

To address the shortcomings described above, the Center for Distributed Object Computing (DOC) and the Applied Research Lab (ARL) at Washington University propose a three year project that will substantially improve core network and middleware technologies available for NGI applications that run over high-speed LANS and WANs. We will accomplish this by integrating and enhancing three proven technologies – WUGS, RIO, and TAO – that we have developed under previous and ongoing government (*e.g.*, NSF and DARPA) and industry (*e.g.*, Bellcore, Boeing, Hughes, Intel, Lockheed, Lucent, Microsoft, Motorola, Nokia, Nortel, SAIC, Siemens, and Sprint) sponsorship.

In our proposed project, WUGS [7] provides the *very high-speed networking infrastructure*, RIO [8] provides a *high-performance, real-time I/O subsystem*, and TAO [9] provides an *open source, standards-based, high-performance and predictable QoS-enabled OO middleware*. In conjunction with targeted DoE collaborators, we will integrate and enhance these synergistic technologies to develop applications that demonstrate the advanced capabilities of our WUGS/RIO/TAO infrastructure configuration. The result will be the first open-source, standards-based, *vertically* (*i.e.*, network interface $\leftrightarrow$ application layer) and *horizontally* (*i.e.*, end-to-end) integrated high-speed network and middleware infrastructure.

1.2 Solution Approach and Expected Results

Below, we outline our plan for leveraging and enhancing our existing WUGS, RIO, and TAO technologies to produce an integrated infrastructure configuration that will have a significant impact on the DoE community. To ensure the success of the proposed project, we will leverage our expertise and ongoing research efforts to produce a *vertically* (*i.e.*, network interface ↔ application layer) and *horizontally* (*i.e.*, end-to-end) integrated configuration infrastructure consisting of the high-speed network, I/O subsystem, middleware, and application components.

Figure 1 depicts the hardware and software components in our proposed WUGS/RIO/TAO integrated architecture. Each of these component is described below:

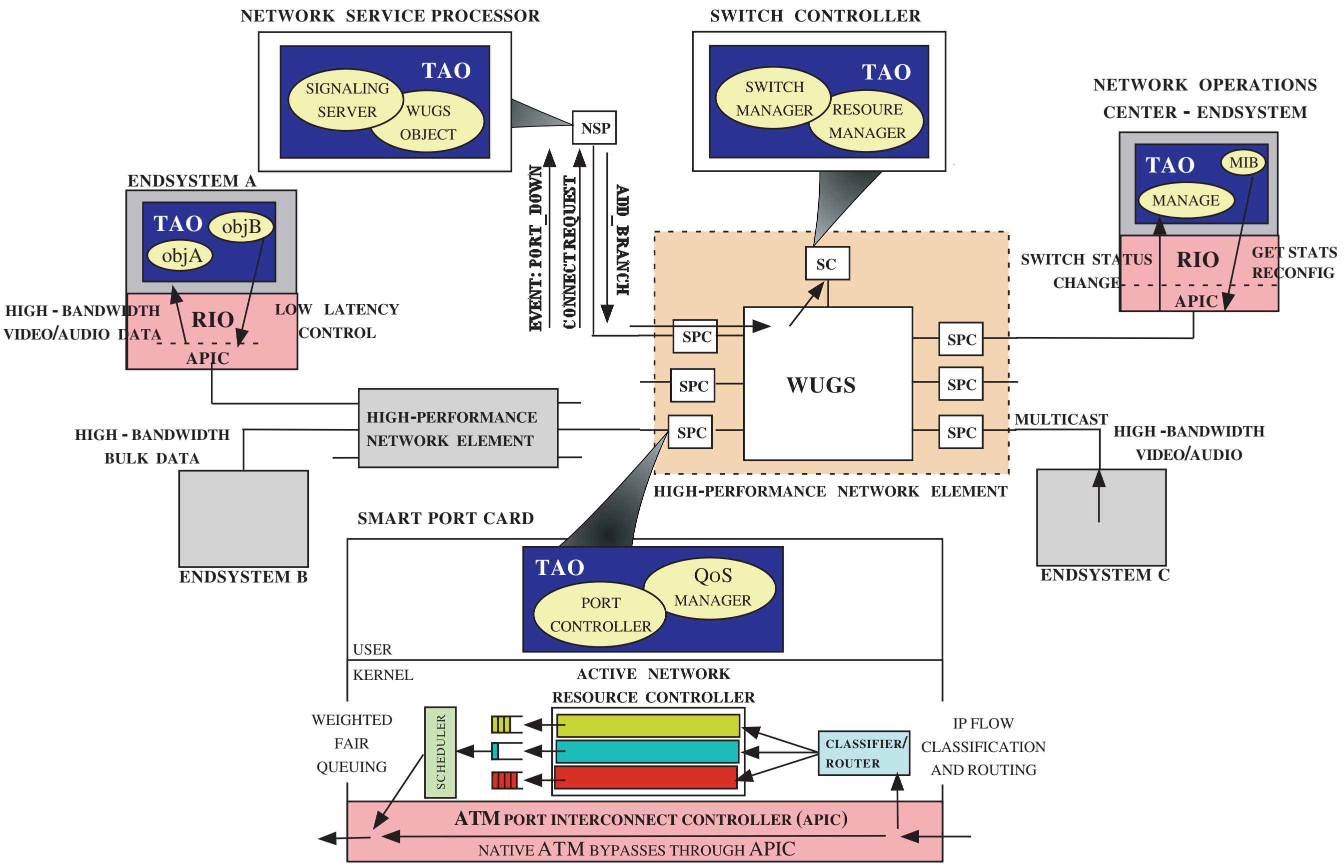

Figure 1: Components in the Integrated WUGS/RIO/TAO Architecture

Washington University Gigabit Switch (WUGS): As shown in Figure 1, the core of our architecture is the Washington University Gigabit Switch (WUGS) [10]. WUGS is a multi-gigabit, QoS-enabled ATM/IP-based networking infrastructure that provides scalable, very high-speed network interfaces. The WUGS-20 switch is built around an eight port switch element with peak throughput of 25 Gbps. Each switch port operates at 2.4 Gbps and can be configured with different combinations of line cards from OC48 to multi-port OC3. Typical configurations include interfaces operating at 1.2 Gbps.

For the proposed effort, we will use the WUGS-20 infrastructure as the basis for our R&D on high-speed, QoS-enabled distributed applications. This will require (1) integrating existing signaling protocols for standards-based network control and (2) developing a end-to-end QoS management and signaling framework within our high-speed network and middleware environment. We will initially support ATM Forum UNI 3.1 and a simple version of ATM PNNI routing. We plan to extend our support to UNI 4.0 using SEAN [11], which is a freely available ATM signaling implementation available from the ATM Signaling Research Group, Center for Computational Sciences, Naval Research Laboratory. SEAN includes a host-native ATM protocol stack and implements the ATM User Network Interface ITU Q.2931 specification, the ITU Q.2971 extension, and the ATM Forum extension UNI-4.0. We will manage ATM switches using an enhanced version of the GSMP [12] protocol.

Real-time I/O (RIO) Subsystem: As shown in Figure 1, endsystems can be outfitted with high-speed network interfaces and an optimized, real-time I/O (RIO) subsystem [8] designed to maximize available bandwidth to a mix of demanding NGI applications. RIO is a high-performance, real-time I/O subsystem currently targeted for a 1.2 Gbps ATM Port Interconnect Controller (APIC) network interface that supports optimized protocol development, zero-copy semantics, and real-time performance [13, 14, 15].

Our existing RIO implementation is targeted at the Solaris environment and includes modified kernel and Fore ATM driver code. This initial prototype does not support (1) zero-copy buffer management, (2) the APIC, or (3) TAO's pluggable protocols framework [16]. Therefore, for the proposed effort we will (1) add support for zero-copy and advanced buffer management strategies, (2) integrate RIO into TAO's pluggable protocols framework, (3) support the APIC without requiring kernel modifications (which will allow open source distribution of RIO), and (4) integrate RIO into other popular OS platforms, such as Windows NT and Linux.

The ACE ORB (TAO): As shown in Figure 1, all applications, network controllers (*i.e.*, signaling processors and network service processors), and embedded network elements (*i.e.*, switch controllers) communicate via TAO [6]. TAO is high-performance, QoS-enabled, standards-compliant [2] middleware that provides a uniform interface for provisioning, monitoring, and controlling the WUGS high-speed network elements. In addition, TAO can be used to develop and deploy high-performance, QoS-enabled NGI applications.

For this proposed effort, we will enhance TAO to (1) provide NGI applications with an IDL interface for specifying QoS requirements, (2) map these QoS requirements to the underlying endsystem I/O subsystems and WUGS network resources and enforcement mechanisms, (3) reduce TAO's memory footprint so it can be embedded into network elements to exploit existing signaling protocols, as well as provide a standards-based signaling component for configuring and managing the communication infrastructure and NGI applications.

Our project deliverables will include (1) a tested and usable integrated infrastructure configuration containing enhanced versions of WUGS, RIO, and TAO, (2) demonstration applications developed in consultation with DoE to showcase the high-bandwidth link speeds and QoS capabilities of our WUGS/RIO/TAO testbed, (3) technical papers; and extensive online documentation in HTML form.

The ARL and DOC Center team has a long history of successfully transferring our state-of-the-art research into tools and products that are used widely in industrial, governmental, and academic projects, such as high-energy physics experiments like BaBar at SLAC [17] and CMS at CERN [18], the HLA/RTI distributed interactive simulation middleware being developed by SAIC under contract with DMSO, as well as avionics mission computing [19], satellite control [20], and electronic medical imaging systems [21, 22]. For this proposed effort, we will leverage our extensive user-base [23] and augment it with a development and deployment process that emphasizes rapid prototyping, periodic demonstrations and course correction evaluations, and coordination with other parts of NGI and other key DoE project participants.

2 Technical Rationale

The fields of high-speed networking, I/O subsystems, and OO middleware research have matured to the point where synergistic collaboration is not only possible, but essential to support NGI applications effectively [24]. Therefore, our proposed integration effort will produce an integrated distributed infrastructure configuration that can preserve the end-to-end QoS guarantees provided by high-speed networks up to a diverse set of NGI applications that are developed, provisioned, monitored, and controlled using flexible, standards-compliant OO middleware. This section summarizes the key research challenges we are addressing and describes how we plan to leverage and enhance our existing expertise to address these challenges for the proposed project.

2.1 High-speed Networking Infrastructure

2.1.1 Challenges

The NGI research community requires robust, flexible high-speed networks that provide comprehensive support for quality of service (QoS) and a variety of communication services, including both ATM and IP, and new services that will be developed in the coming years. These services should be accessible through platform-independent middleware APIs to facilitate application development and deployment across the widest possible range of operating environments. The following challenges must be met to enable new network technologies to deliver the services and applications needed by NGI users and ultimately, the public at large.

1. Multi-service routing switches: These switches must provide a range of communication services, including ATM, IP (over ATM, SONET, Ethernet), enhancements to these services (signaling, flexible and scalable multicast services, security services) and dynamic provision of new services through programmable network elements. Such systems will require a scalable switch fabric, flexible processing capabilities at each switch port and a range of physical interface options.

2. Distributed control software for scalable multi-service routing switches: As routing switches expand to include tens, and even hundreds of high-speed ports, new control system architectures are needed to manage system resources effectively, ensuring essential coordination while minimizing the interactions required among distributed processing elements.

3. Network services for burst-level bandwidth management: Network-level mechanisms, such as ABR flow control, can provide faster and more precise control of network bandwidth usage than transport-level mechanisms, such as TCP's adaptive windowing. However, both depend on burst durations that are much longer than network round-trip times, providing little support for the increasingly common case of applications that must send large amounts of data in relatively short time intervals.

4. QoS-enabled queuing mechanisms: These mechanisms must support fine-grained bandwidth assignment and differential treatment of reservation-oriented and bandwidth-adaptive traffic flows. Experimentation with advanced queuing mechanisms in high-speed networks is needed to assess costs and benefits in a realistic network context.

5. Network support for large-scale multicast applications: A growing number of NGI applications are expected to take advantage of multicast services [25]. Network-level support for error and flow control can provide the scalability needed to enable applications expected to involve hundreds or even thousands of participants. Many-to-many multicast applications raise basic definitional problems for QoS since transmissions by different endpoints are often strongly correlated.

2.1.2 Leveraged Technology

To meet the challenges presented by the NGI, researchers require a flexible and open research platform on which to build and experiment. Commercial switch and router vendors cannot provide the access needed to support the DoE research community. Likewise, the alternative of using workstations as experimental routers greatly limits the performance range of resulting systems. These constraints make it hard to use commercial networking elements in demanding testbed environments and prevents researchers from grappling with key issues associated with systems supporting large numbers of high-speed ports.

To address these problems, Washington University has developed a scalable ATM gigabit switch (WUGS) and associated technology that can be used to construct flexible, high-performance routing switches. The WUGS technology is being distributed to over 30 universities and companies in an open and extensible research platform called *Gigabit Network Kits* [26]. A local company, STS Technologies [27], is producing the systems commercially, under an agreement that maintains the open nature of the WUGS architecture. All technical details, including internal interfaces and even integrated circuit logic definitions are available to participants, allowing them to develop new software and hardware extensions that build upon and leverage the existing base.

The distributed technology package includes a high-performance ATM switch core (the WUGS-20) [10], line cards for different physical interfaces, and a high-performance network interface card based on a dual-port network interface chip called the APIC (ATM Port Interconnect Controller) [28, 29]. Current developments include a (1) Smart Port Card (SPC) [30] that enables a high-performance embedded processor to be installed at each switch port and (2) a set of OS extensions that allow dynamic configuration of kernel plug-ins within this embedded processor. The kernel plug-in technology is being used to support dynamic configuration of new network and application services as part of an on-going research program in active networking [31].

Under an existing effort, the Applied Research Laboratory is developing a larger configuration of the WUGS-20 called the WUGS-160 [10]. The new WUGS-160 will provide an aggregate capacity of up to 160 Gbps and is designed to support the same kind of extensions to both hardware and software as the smaller WUGS-20s. In addition, we are planning the next-generation of WUGS, which will add a configurable hardware port card based on a high-performance FPGA. This feature will be used to prototype advanced IP address lookup and packet classification algorithms and can also be used to implement QoS queuing mechanisms. While the proposed effort plans to use the existing WUGS-20, the DoE testbed can be expanded to incorporate this newer technology in the future.

2.1.3 Innovative Technology

Multi-service routing switches require both an effective and flexible hardware platform, and middleware software mechanisms to manage system resources flexibly and reliably. The WUGS switch and ancillary hardware components provide an effective base on which to build an experimental multi-service routing switch. Therefore, our proposed project will integrate the various hardware and software components, as well as provide the overall system management and signaling support needed to coordinate activities across all the different components of the system.

We plan a scalable system architecture in which most per-port control functions are provided by embedded processors implemented using Smart Port Cards (SPC)s. As shown in Figure 2, the SPCs used for IP forwarding, per port QoS enforcement and active network nodes are labeled as Port Processors (PP) and are shown as being attached to each switch interface. The individual PPs are configured by a central Switch Controller (SC), which provides overall coordination of hardware resources. The SC will communicate with individual PPs using TAO and will configure the switch forwarding tables using control cells sent to the appropriate ports. Likewise, the PPs will send event notifications and replies to the SC using TAO.

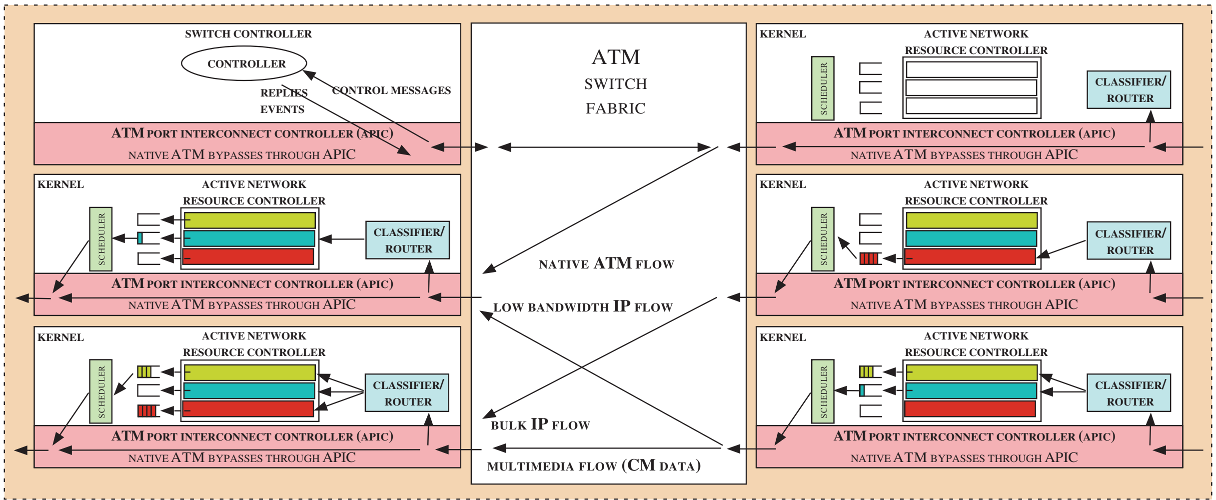

Figure 2: WUGS with Embedded Port Processors

The SC will export a CORBA IDL-based switch interface using the General Switch Management Protocol (GSMP) [12] and/or the Multiservice Switching Forum's Virtual Switch Interface [32] using TAO. All communication with the SC will be using CORBA and TAO specifically. Signaling Processors (SP) will coordinate the setup of ATM virtual circuits and reserved IP flows using both standard and experimental signaling protocols. The SP interactions with the switch hardware will be mediated by the SC, which will arbitrate competing resource demands to ensure effective overall use of system resources.

By using TAO and RIO in conjunction with the SC, SP and PP we will provide (1) a consistent, extensible control interface and (2) QoS guarantees for the control messages themselves. Our proposed research activities for TAO and RIO, described below, will use the WUGS high-speed networking infrastructure for R&D on active networks, differentiated servers, signaling, QoS management, network management, and other topics related to developing performance-sensitive NGI applications.

Section 3.1 presents the specific activities associated with developing our high-speed networking infrastructure based on innovations to the leveraged WUGS technology described above.

2.2 High-performance and Real-time I/O Subsystem

2.2.1 Challenges

Meeting the requirements of distributed real-time applications requires more than defining QoS interfaces with CORBA IDL or developing an ORB with real-time thread priorities [33, 34]. In particular, it requires the integration of the ORB, OS I/O subsystem, and high-speed network elements in order to provide end-to-end QoS-enabled scheduling and communication to the ORB endsystem. The following research challenges confront conventional computer architectures, operating systems, and network interfaces, which cannot process I/O requests at contemporary network link speeds [35, 36, 37, 38]:

1. System bus bottleneck: In theory a PCI bus can burst 1,056 Mbps (33 MHz, 32 bit) up to 4,224 Mbps (66 MHz, 64 bit). In practice, however, the actual throughput is somewhat less, *e.g.*, 720 Mbps to 2,880 Mbps, since burst-mode DRAM requires 2 or more bus cycles for the first word of the burst. Interfaces that support direct read/write operations to system memory and protected mode DMA [29] are required to reduce the overall load on the system bus.

2. Interrupt handling of I/O events: A prime source of priority inversion is the processing of I/O events in interrupt context. Processing received data in interrupt context can produce *receive livelock* [39], which yields potentially unbounded priority inversion. To avoid this problem, high-performance I/O subsystem's must support mechanisms to enforce the prioritized handling of I/O requests.

3. Lack of zero-copy interfaces for network I/O operations: In conventional ORB endsystems, the data received from a network interface may be copied multiple times, along with an additional data read to perform checksum operations [40]. As a result, one network I/O operation may incur multiple bus and memory operations, which substantially reduces the maximum achievable throughput. Thus, mechanisms are necessary to support zero-copy I/O semantics between network interfaces, the OS I/O subsystem, and higher-level middleware and applications.

4: Unintegrated filesystem and network I/O subsystems: In conventional operating systems, networked file I/O crosses the user-kernel protection boundary at least twice when accessing a filesystem through the network. This results in excessive data copies and inefficient use of system buffer space, which reduces overall system capacity [41] and limits the resources available for application programs. These resources, CPU and Buffer space) are squandered copying data from filesystem buffers to the user application and then to the network I/O subsystem An integrated approach is needed to manage network and filesystem buffers that allow direct transfer of data between I/O subsystems without unnecessary data copies.

5. Lack of a QoS API for I/O subsystems: Developers require a standard API to specify application QoS requirements to the underlying I/O subsystem. Even if mechanisms are available for controlling I/O resource allocation, they may be too complex to use and may vary dynamically with system loading. Common mechanisms for allocating and enforcing I/O resources are dedicated kernel threads for I/O operations, setting maximum interrupt rates, periodic interface polling, zero-copy interfaces, advanced buffering strategies and interface bandwidth control. In addition, there are various approaches for providing feedback to applications about their current resource usage. What is needed is a standard interface for QoS specification and for providing feedback on current performance [37, 42].

6. Packet-based priority inversions: In conventional I/O subsystems, incoming packets are processed in FIFO order [8]. This FIFO policy can result in head-of-line blocking, where packets destined for high-priority threads must wait for protocol processing to finish on packets destined for lower priority threads. Packet-based priority inversion can be alleviated by performing early demultiplexing of received packets [8] and maintaining a set of prioritized or weighted input queues [43].

2.2.2 Leveraged Technology

Our prior work on high-performance and real-time I/O subsystems [14, 44, 45] identified limitations with conventional I/O subsystems and network interfaces. We have addressed many of these limitations, such as receive livelock, early demultiplexing of requests, prioritized protocol processing, and vertical integration of request processing, in our RIO project [8]. Our other I/O subsystem work has focused on multimedia applications, services, enforcement and interfaces.

For example, a high-performance ATM Port Interface Controller [28] interface has been designed and fabricated. We have also prototyped complementary technology, such are virtual memory and zero-copy buffer semantics [44, 41].

We have applied our I/O subsystem technology to high-performance endsystems and desk area networks [14], focusing particularly on multimedia endsystem design [46, 47]. A novel CPU scheduling scheme called *rate monotonic with delayed preemption* (RDMP) [48] was developed to support the periodic scheduling of continuous media I/O. In this work, RDMP was used as the scheduling component of a unique upcall mechanism, called real-time upcalls (RTUs), which perform protocol processing in user space [45]. RTUs were applied in the multi-media server developed in project MARS [49], which provides QoS guarantees within the filesystem and data on disk.

Figure 3 show how our real-time I/O (RIO) subsystem has been implemented in Solaris [50] using the leveraged technologies described above. The key optimizations used in our RIO subsystem to maximize efficiency and provide

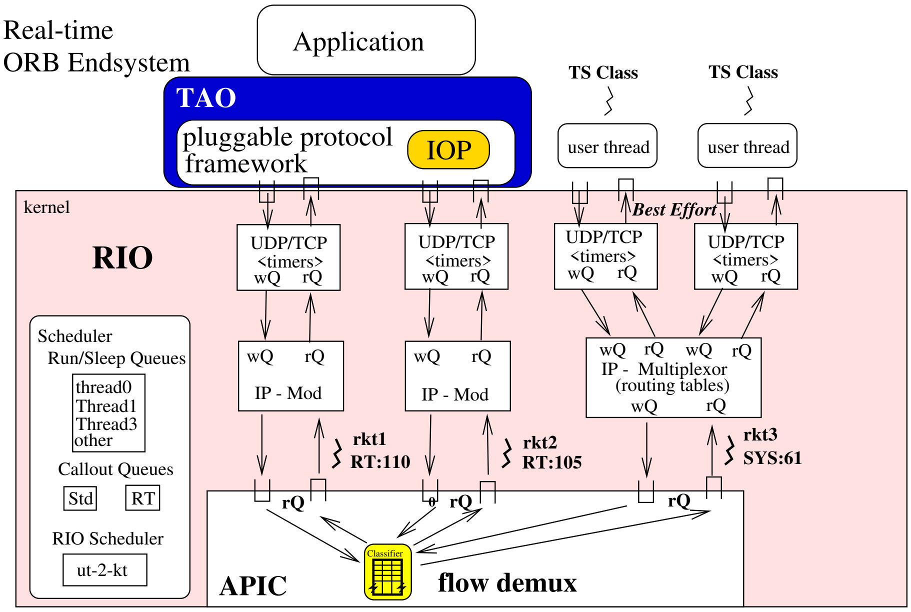

Figure 3: Components in the RIO Subsystem

QoS guarantees include: (1) high-speed network interfaces that support advanced I/O features, such as protected DMA, read/write directly to host memory, priority queues, programmable interrupts and paced transmission, (2) data demultiplexing at the lowest level to ensure service/performance isolation between various active applications, (3) vertically integrating software layers to increase efficiency and avoid unnecessary (de)multiplexing, (4) binding an appropriately-prioritized periodic task/thread to each data flow to allow QoS guarantees for protocol and filesystem processing, (5) using a zero-copy buffer management system that works generally, *e.g.*, with the filesystem and network protocol stack, to eliminate unnecessary data-copying overheads, and (6) processing I/O events according requested or imposed QoS attributes, and not on demand when packets are received.

2.2.3 Innovative Technology

High-performance, QoS-enabled ORB endsystems need both an optimized I/O subsystem and a programming API to allow applications to exploit this capability via standardized middleware. In our proposed effort, therefore, we will port the innovative RIO subsystem technologies described above from Solaris and NetBSD (where it currently runs) to popular PC platforms, such as Windows NT and Linux. In addition, we will enhance the RIO subsystem to include support for the APIC and other high-speed network interfaces, such as those conforming to the VIA [51] standard.

The primary vehicle for integrating the enhanced RIO subsystem into the underlying high-speed WUGS network infrastructure and the higher-level QoS-enabled middleware is TAO's *pluggable protocols framework*, which is shown in Figure 3 and described in Section 2.3.3. This framework provides an intuitive and powerful protocol configuration API that encapsulates the RIO subsystem and network elements within the TAO ORB. This design allows new and legacy software systems to benefit from our high-performance, QoS-enabled I/O subsystem.

Section 3.2 presents the specific activities associated with developing our high-performance and real-time I/O subsystem based on innovations to the RIO subsystem technology described above.

2.3 High-Performance, QoS-enabled Middleware

2.3.1 Challenges

Performance-sensitive distributed applications, such as collaborative multimedia teleconferencing, testbeam data acquisition, and interactive programming steering applications, have historically been developed using non-standard programming APIs due to the following shortcomings of conventional off-the-shelf middleware, such as CORBA, DCOM, or Java RMI [6]:

Lack of QoS specification interfaces: The CORBA 2.x standard does not provide interfaces to specify end-to-end QoS requirements. For instance, there is no standard way for clients to indicate the relative end-to-end priorities of their requests to an ORB. Likewise, there is no interface for clients to inform an ORB the rate at which to execute operations that have periodic processing deadlines.

Lack of QoS enforcement: Conventional ORBs do not provide end-to-end QoS enforcement, *i.e.*, from application-to-application across a network. For instance, most ORBs transmit, schedule, and dispatch client requests in FIFO order. However, FIFO strategies can yield unbounded priority inversions [52, 53], which occur when a lower priority request blocks the execution of a higher priority request for an indefinite period. Likewise, conventional ORBs do not allow applications to specify the priority of threads that processes requests.

Lack of performance optimizations: Conventional ORB endsystems incur significant throughput [54] and latency [55] overhead, as well as exhibiting many priority inversions and sources of non-determinism [8]. These overheads stem from (1) non-optimized presentation layers that copy and touch data excessively [56] and overflow processor caches [57]; (2) internal buffering strategies that produce non-uniform behavior for different message sizes [58]; (3) inefficient demultiplexing and dispatching algorithms [59]; (4) long chains of intra-ORB virtual method calls [54]; and (5) lack of integration with underlying real-time OS and network QoS mechanisms [33, 6, 8].

Application developers typically address these limitations by either (1) relying on "best effort" delivery semantics in underlying networks and I/O subsystems or (2) crafting domain-specific middleware and programming APIs. Neither of these solutions scale very well to meet the requirements of complex NGI applications. For example, both ATM [60] and RSVP [61] provide end-to-end network resource allocation mechanisms. However, if applications want to use either ATM or RSVP to specify and enforce their QoS, they would first have to acquire and possibly port the libraries to their platform. Next, they would have to program to an RSVP-specific API, such as WinSock2 or XTI. Unfortunately, these APIs are tedious, error-prone, and non-portable, which makes it hard to share expertise, software infrastructures, and applications across DoE projects.

What is needed, therefore, is a more general, open standards-based middleware framework [24] that incorporates common protocol optimizations, real-time features, and enhanced QoS specification interfaces and enforcement mechanisms to seamlessly integrate with advances in network and endsystem research. This middleware framework can then be used to supply a standard API for QoS specification/enforcement, real-time programming features, and performance optimizations. Moreover, such a middleware framework must also support *custom* mechanisms and protocols.

2.3.2 Leveraged Technology

To address the limitations with existing middleware described above, we have developed The ACE ORB (TAO) [9]. TAO is open-source, standards-based, high-performance, real-time ORB endsystem middleware that supports applications with deterministic and statistical QoS requirements, as well as "best-effort" requirements. TAO is the first ORB to support end-to-end QoS guarantees over ATM/IP networks [62, 8].

Under sponsorship from government (*i.e.*, NSF and DARPA) and industry (*i.e.*, Bellcore, Boeing, Hughes, Lockheed, Lucent, Microsoft, Motorola, Nokia, Nortel, SAIC, Siemens, and Sprint), we have developed the features and optimizations shown in Figure 4 and described below:

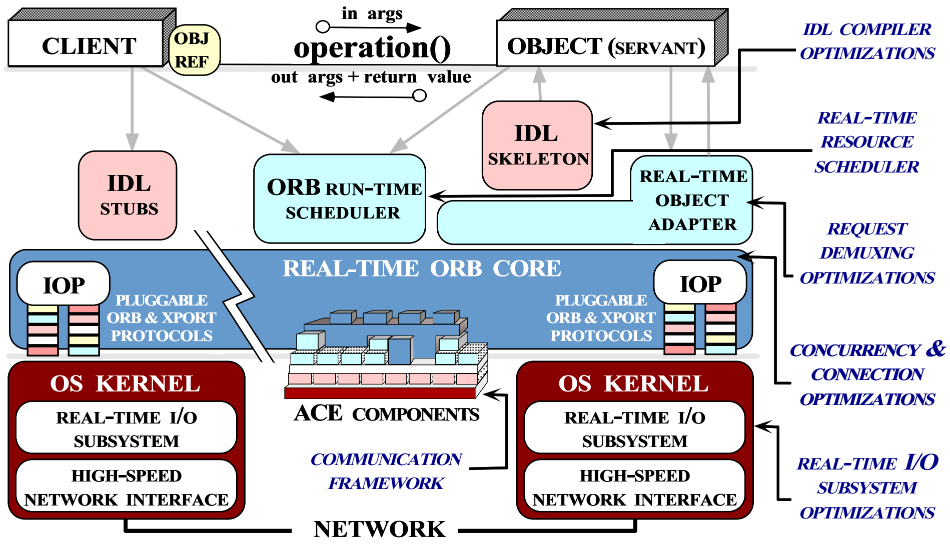

Figure 4: Components in the TAO Real-time ORB Endsystem

1. Highly scalable and predictable request demultiplexing strategies: TAO's Object Adapter demultiplexing strategies route requests to objects in constant, *i.e.*, O(1), time regardless of the number of objects and IDL interface operations [50].

2. IDL compiler optimizations: TAO's IDL compiler can generate optimized compiled and/or interpreted stubs and skeletons, which enable applications to make fine-grained time/space tradeoffs at the presentation layer [57].

3. Efficient and predictable concurrency and connection models: TAO's ORB Core concurrency and connection models minimize context switching, synchronization, dynamic memory allocation, and data movement and avoid priority inversion and behave predictably when used with multi-rate, real-time applications [63]. TAO's ORB Core also allows customized transport protocols [16] to be plugged into the ORB without affecting the standard CORBA application programming model.

4. A real-time I/O subsystem: TAO can be configured to leverage the real-time I/O (RIO) subsystem prototype described in Section 2.2.2. RIO minimizes priority inversion interrupt overhead over high-speed network interfaces, such as the WUGS ATM switches or Gigabit Ethernet [8]. TAO also runs efficiently and relatively predictably on conventional I/O subsystems that lack advanced QoS features.

5. A real-time resource scheduler: TAO's scheduler maps application QoS requirements, such as end-to-end latency or periodic deadlines, to ORB endsystem and network resources, such as CPU, primary and secondary storage, and network bandwidth [6, 64].

6. A highly portable communication framework: TAO achieve a high-degree of portability across platforms, TAO is developed using the ACE framework [65], which implements reusable C++ wrapper facades and framework components that support the QoS requirements of high-performance, real-time applications and higher-level middleware like TAO. ACE and TAO run on most OS platforms, including supercomputer operating systems, such as Cray UNICOS, real-time operating systems, such as Chorus, LynxOS, and VxWorks, and general-purpose operating systems with real-time enhancements, such as Windows NT, Solaris, and Linux.

2.3.3 Innovative Technology

To build a robust, efficient, and scalable OO middleware framework the network element controllers, endsystems, and network operations centers must all support, and be supported by, the higher-level middleware. Therefore, we will extend existing TAO ORB middleware to support the following new features:

QoS-enabled middleware: We will define a QoS API to allow NGI applications to specify their QoS requirements using CORBA IDL interfaces. We will then define a mapping of these application QoS specifications to the underlying WUGS network and RIO subsystem mechanisms by leveraging and enhancing TAO's pluggable protocols framework [16]. Figure 5 shows the partitioning of responsibilities for pluggable protocols and how it relates to other ORB and networking services.

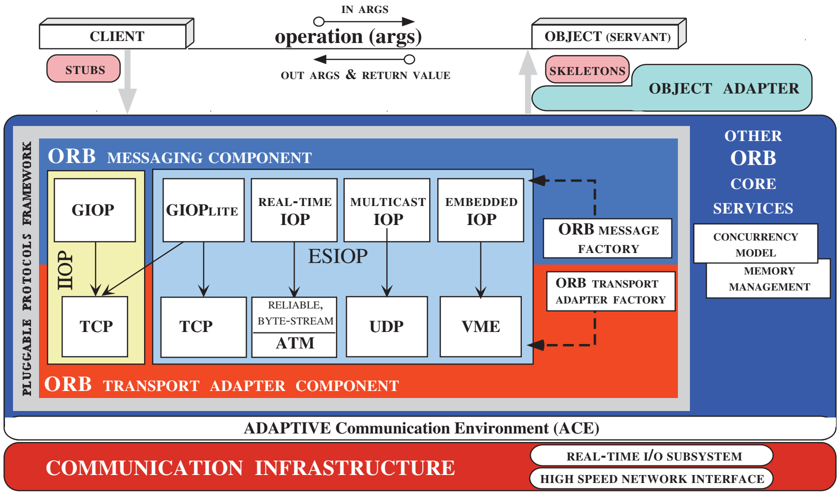

Figure 5: TAO's Pluggable Protocols Framework Architecture

When complete, TAO's QoS-enabled pluggable protocols framework will allow custom ORB messaging and transport protocols to be configured flexibly and used transparently by NGI applications. For example, if ORBs communicate over high-speed networking protocols with QoS support, such as WUGS ATM/IP, simpler, optimized ORB messaging and transport protocols can be configured to eliminate unnecessary features and overhead of the standard CORBA General Inter-ORB Protocol (GIOP) and Internet Inter-ORB Protocol (IIOP). Likewise, TAO's pluggable protocols framework makes it straightforward to support customized embedded system interconnects, such as CompactPCI or VMEBus, under standard CORBA inter-ORB protocols like GIOP.

TAO's ORB transport layer will be integrated with RIO and WUGS to include advanced I/O buffer management strategies that configure and control the I/O subsystem advanced mechanisms. For example, this layer will implement zero-copy buffer management schemes that leverage the APIC high-speed network interface described in Section 2.1.2. TAO's pluggable protocol framework will leverage other aspects of the WUGS high-speed networking infrastructure. For instance, TAO's QoS specification APIs will allow applications to program Smart Port Cards (SPC)s, which can help enforce QoS guarantees end-to-end.

Embedded TAO: We will make TAO compliant with the new Minimum CORBA specification [66] to create a small footprint ORB that can be embedded into WUGS network switches, as well as associated control processors and signaling processes. By using embedded TAO, NGI applications will be able to transparently control end-to-end flows with associated QoS guarantees. The embedded TAO ORBs in the network elements will communicate using either custom or standard Inter-ORB Protocols (IOP)s. Thus, QoS-related signaling can be performed by simply invoking standard CORBA remote operation calls, rather than having to program directly to proprietary lower-level messaging protocols.

A snapshot of our complete QoS-enabled middleware framework is presented in Figure 6. In this framework we will have ORBs (1) embedded on the switches (*e.g.* switch controllers) and (2) resident on the client hosts and the signaling processors. This framework will provide an open switch/router control framework that can provision end-to-end QoS guarantees for real-time and high-bandwidth applications running over high-speed networks, such as ATM, Gigabit Ethernet, and high-speed IP routers.

Section 3.3 presents the specific activities associated with developing our high-performance, QoS-enabled middleware based on innovations to the TAO technology described above.

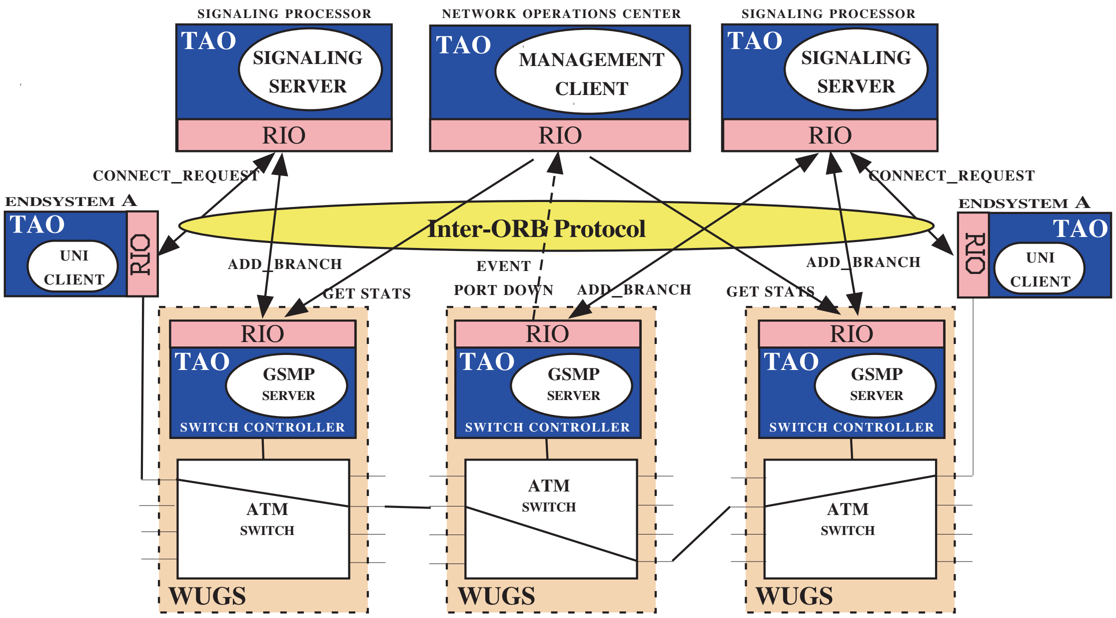

Figure 6: Components in the TAO QoS-enabled Middleware Framework

2.4 Demonstration of Our Integrated Testbed

In consultation with DoE, we will develop an application similar to the one shown in Figure 7. This application will support groups of scientists collaborating over high-speed networks, downloading large multimedia data sets, interacting remotely using visual displays, spawning analysis computations dynamically, monitoring and controlling long-running computations, and updating/refining the data in a database, as well a serve as a vehicle for empirical testing and evaluation.

Our collaborative application will showcase the advanced QoS control capabilities of the integrated WUGS/RIO/TAO configuration infrastructure. These capabilities will allow the application and/or administrators to manipulate a variety of network control parameters in response to observed behavior and to specify policies to suit specific requirements. This will include vertically and horizontally control mechanisms at the network (*e.g.*, ATM and IP), I/O subsystem and middleware (*i.e.*, RIO and TAO, respectively), and application levels.

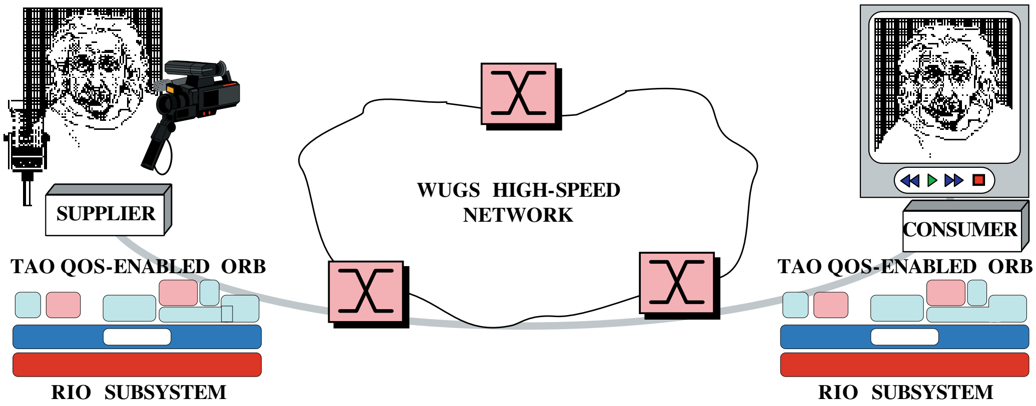

Figure 7: Collaborative Application Running Over the Integrated WUGS/RIO/TAO System

At the WUGS ATM level, we will control parameters governing admission control, routing and packet level discarding. At the IP level, we will include control of parameters for packet queuing and packet or flow routing. At the RIO subsystem and TAO middleware level, we will control interface services that can detect resource contention at run-time and react dynamically to improve the amount and type of data that can be delivered while maintaining end-to-end real-time application timing constraints. At the application level, we will optimize WUGS/RIO/TAO adaptive feedback loop to minimize latency to a point representative for these types of collaborative multimedia applications.

Section 3.4 presents the specific activities associated with developing our high-speed networking infrastructure based on innovations to the WUGS technology described above.

3 Statement of Work

The proposed project is a three year effort that consists of four main tasks, each containing several subtasks, as described in this section.

3.1 Task 1 – Integrate and Enhance the WUGS High-speed Networking Infrastructure

This task will provide a multi-gigabit networking infrastructure, high-performance IP routers, active network nodes and ATM switches that will be used in subsequent tasks to develop NGI applications and conduct performance experiments and integrated demonstrations. Task 1 is comprised of the following subtasks:

Task 1.1: Enhance the WUGS-20 network testbed to use the next-generation WUGS-160 switches, which support 64 OC-48 links for an aggregate throughput of 160 Gbps.

Task 1.2: Develop and integrate Smart PortCards (SPCs) into WUGS to monitor traffic and process IP packets at gigabit rates [62, 67].

Washington University's Applied Research Laboratory (ARL) has extensive expertise in these areas. In particular, ARL's WUGS *Gigabit Network Kits* [26] are currently being distributed to over 30 universities and companies. Along with kits, researchers are provided detailed information about the design and operation of the kit components. There is also a program for researchers to contribute developed hardware and software back into the program. These kits are being used to research various topics including cluster computing, high speed IP routers and network management.

3.2 Task 2 – Integrate and Enhance the RIO High-performance and Real-time Endsystem I/O

This task will integrate high-speed network elements, endsystem network interfaces, and middleware by incorporating the APIC and applicable high-performance I/O techniques into TAO's pluggable protocol framework on popular platforms, such as Windows NT, Linux, LynxOS, or NetBSD. Task 2 is comprised of the following subtasks:

Task 2.1: Add ORB level support for high-performance interfaces, like the APIC, within the pluggable protocols framework. This includes buffer management for zero-copy I/O and adding interfaces to control network driver level features like interrupt rate, polling, early demultiplexing and processing priorities.

Task 2.2: Integrate the RIO subsystem into TAO's pluggable protocols framework.

Task 2.3: Port APIC driver and develop support libraries for popular operating systems such as Windows NT, Linux, and LynxOS.[1]

Task 2.4: Add zero-copy semantics and advanced buffer management strategies ensure the resulting driver and libraries are compatible with high-performance operating systems, such as Cray UNICOS; real-time operating systems such as Chorus, LynxOS, and VxWorks; and general-purpose operating systems with real-time enhancements, such as Windows NT, Solaris, and Linux.

Task 2.5: Implement advanced I/O subsystem features such as early demultiplexing, vertically integration (*i.e.*, isolated data paths with reserved resources), periodic protocol processing, interrupt rate control, polling, prioritized queues and output pacing to the aforementioned popular platforms.

As described in Section 2.2.2, Washington University's Center for Distributed Object Computing and Applied Research Laboratory has extensive experience in real-time I/O subsystems, multi-media servers and high-performance networking. In Task 2, we will leverage this experience and our existing I/O subsystem implementations to produce an integrated high-performance, real-time I/O subsystem (RIO) communications infrastructure that leverages WUGS and provides end-to-end QoS enforcement for TAO.

[1] Current APIC driver development is targeted to NetBSD.

3.3 Task 3 – Integrate and Enhance the TAO High-Performance, QoS-enabled ORB Middleware

This task will focus on enhancing the existing TAO framework to included support for generic QoS specification, QoS enforcement for specific networking technologies and protocols, produce a minimal footprint ORB for embedding in network switches and develop optimal Inter-ORB protocol implementation for this environment and within the TAO pluggable protocol layer, integrate with existing signaling protocols. Task 3 is comprised of the following subtasks:

Task 3.1: Define IDL interfaces that map application-level QoS requirements to the underlying network and OS mechanisms, using pluggable protocols as an enabling technology.

Task 3.2: Integrate TAO's pluggable protocol framework into the WUGS high-speed network testbed and RIO subsystem.

Task 3.3: Make TAO comply to the Minimum CORBA specification [66] and embed it in the switches (*e.g.*, switch controllers) and on the client hosts and the signaling processors to provide an open switch/router control framework that can provision end-to-end QoS guarantees for real-time and high-bandwidth applications running over high-speed networks, such as ATM, Gigabit Ethernet, and high-speed IP routers.

Task 3.4: Implement the selected signaling protocol(s) using embedded TAO.

Washington University's Center for Distributed Object Computing (DOC) and Applied Research Laboratory (ARL) have extensive experience in these areas. As outlined in Section 2.3.2 the DOC Center has developed TAO, which is a highly optimized, real-time CORBA 2.3-compliant Object Request Broker (ORB) [9]. Likewise, ARL has conducted extensive work on lightweight integrated signaling protocols for establishing resource reservations. This effort included research on IP switching as part of the Crossbow project [68, 69], which (1) demonstrated a Cell-switched Router using ATM switches with Pentium PCs as control processors and (2) developed a signaling protocol, called the *State Setup Protocol* (SSP) [70], that establishes QoS bindings and allocates VCs.

3.4 Task 4 – Demonstration of a Collaborative Application and Results of Empirical Studies in Our Integrated Testbed

In this task, we will develop and conduct systematic performance benchmarks using synthesized traffic patterns and actual applications to explore the design space of QoS-sensitive parameters in our integrated testbed. Task 4 is comprised of the following subtasks:

Task 4.1: In consultation with DoE collaborators, capture requirements to guide the design and implementation of a collaborative application to showcase the advanced QoS feedback control capabilities of the integrated WUGS/RIO/TAO configuration infrastructure. An example application is illustrated in Figure 7 and described in Section 2.4.

Task 4.2: Develop the application using IDL interfaces and CORBA. Developing this application will help direct our focus to representative problem areas with WUGS/RIO/TAO that require additional attention.

Task 4.3: Assemble and test a prototype WUGS/RIO/TAO testbed built around a 2-3 node WUGS-20 switching fabric and 16 OC-12 interface cards.

Task 4.4: Experimentally evaluate the testbed with synthesized traffic to help explore portions of the design space that are not exercised by the applications.

Task 4.5: Using the feedback from earlier tasks, enhance the functionality and performance of WUGS, RIO, and TAO.

Washington University's Center for Distributed Object Computing (DOC) and Applied Research Laboratory (ARL) have extensive experience developing multimedia applications. For instance, the DOC Center has developed a CORBA-based Audio/Video Streaming Service [71]. Likewise, ARL and the DOC Center have developed a high-speed network management framework [72] that allow end-users, applications, and administrators to monitor, visualize, and control the performance of their end-to-end QoS. Moreover, ARL has developed Vaudeville, which is a voice-activated teleconferencing application. It supports simultaneous multiple multi-party conferences where users are free to dynamically add and remove membership in different teleconference sessions.

4 Related Work

4.1 Related Work on High-speed Network Infrastructures

The last decade has seen the development of a number of high-performance network testbeds, intended to support networking research. Currently, the National Science Foundation's vBNS [73] network offers the largest example of such a high-speed network testbed. The vBNS now spans several tens of sites (including Washington University) and provides an effective mechanism for delivering higher bandwidth to application researchers. Its value to networking researchers has been more limited, however, since the network provides only standard IP datagram services and many aspects of the networks operation are largely off-limits to networking researchers. This makes vBNS unsuitable as an environment in which to develop, deploy, and evaluate experimental services. It is also relatively low performance by emerging standards, *e.g.*, for cost reasons, most institutions access to the network is limited to 45 Mbps and only a small handful have connections of more than 150 Mbps.

CAIRN [74] is a national testbed for research in Internet protocols and technology. CAIRN now connects dozens of sites at speeds from 1.5 Mbps up to 150 Mbps. CAIRN is built using a common experimental routing platform, which allows researchers to experiment with new network services by developing and loading new software for the router. The current routing platform is limited in performance since it is built around a single general-purpose computer with multiple network interface cards. The architecture is comparable to commercial routers of the mid-1980s, but lags far behind the current state-of-the-art. This makes it difficult for CAIRN researchers to tackle many of the performance-related challenges that are at the cutting edge of current research.

The National Transparent Optical Network (NTON) [75] is a network testbed now in the planning stages that will span a number of institutions along the west coast of the U.S. NTON planners are attempting to structure the testbed to support both applications researchers and networking researchers operating at different levels of the network hierarchy. If they can in fact develop mechanisms to enable network and application researchers to effectively co-exist in a common testbed infrastructure, network researchers will have the opportunity to develop and deploy new network services and evaluate them in the context of a high-performance testbed with a substantial user population. Realizing this vision will require inclusion of experimental network equipment that is open to modification and extension.

4.2 Related Work on High-performance and Real-time I/O Subsystems

Our real-time I/O (RIO) subsystem [8] incorporates advanced techniques [35, 29, 36, 38, 76] for high-performance and real-time protocol implementations. Our RIO work is based on an earlier effort which looked at implementing networking protocols in user space and providing end-to-end QoS guarantees for multimedia applications [43]. This work also relied on early demultiplexing, periodic protocol processing [45] with guarantees and prioritized protocol processing. A new scheduling policy RMDP was employed to limit the cost of context switching and to provide predictable scheduling. The RIO work built on these ideas and moved them from the NetBSD environment to a multi-threaded, preemptive kernel environment with the protocol processing moved back into the kernel.

I/O subsystem support for QoS: The Scout OS [77, 78] employs the notion of a *path* to expose the state and resource requirements of all processing components in a *flow*. Similarly, our RIO subsystem reflects the path principle and incorporates it with TAO to create a vertically integrated real-time ORB endsystem. For instance, RIO subsystem resources like CPU, memory, and network interface and network bandwidth are allocated to an application-level connection/thread during connection establishment, which is similar to Scout's binding of resources to a path.

Scout represents a fruitful research direction, which is complementary with our emphasis on demonstrating similar capabilities in existing operating systems, such as Solaris and NetBSD [44]. At present, paths have been used in Scout largely for MPEG video decoding and display and not for protocol processing or other I/O operations. In contrast, we have successfully used RIO for a number of real-time avionics applications [19] with deterministic QoS requirements.

SPIN [79, 80] provides an extensible infrastructure and a core set of extensible services that allow applications to safely change the OS interface and implementation. Application-specific protocols are written in a typesafe language, *Plexus*, and configured dynamically into the SPIN OS kernel. Because these protocols execute within the kernel, they can access network interfaces and other OS system services efficiently. To the best of our knowledge, however, SPIN does not support end-to-end QoS guarantees.

Enhanced I/O subsystems: Other related research has focused on enhancing performance and fairness of I/O subsystems, though not specifically for the purpose of providing real-time QoS guarantees. These techniques are directly applicable to designing and implementing real-time I/O and providing QoS guarantees, however, so we compare them with our RIO subsystem below.

[38] applies several high-performance techniques to a STREAMS-based TCP/IP implementation and compares the results to a BSD-based TCP/IP implementation. This work is similar to RIO since they parallelize their STREAMS implementation and implement early demultiplexing and dedicated STREAMS, known as Communication Channels (CC). The use of CC exploits the built-in flow control mechanisms of STREAMS to control how applications access the I/O subsystem. This work differs from RIO, however, since it focuses entirely on performance issues and not sources of priority inversions. For example, minimizing protocol processing in interrupt context is not addressed.

[39, 36] examines the effect of protocol processing with interrupt priorities and the resulting priority inversions and livelock [39]. Both approaches focus on providing fairness and scalability under network load. In [36], a network I/O subsystem architecture called *lazy receiver processing* (LRP) is used to provide stable overload behavior. LRP uses early demultiplexing to classify packets, which are then placed into per-connection queues or on network interface channels. These channels are shared between the network interface and OS. Application threads read/write from/to network interface channels so input and output protocol processing is performed in the context of application threads. In addition, a scheme is proposed to associate kernel threads with network interface channels and application threads in a manner similar to RIO. However, LRP does not provide QoS guarantees to applications.

[39] proposed a somewhat different architecture to minimize interrupt processing for network I/O. They propose a polling strategy to prevent interrupt processing from consuming excessive resources. This approach focuses on scalability under heavy load. It did not address QoS issues, however, such as providing per-connection guarantees for fairness or bandwidth, nor does it charge applications for the resources they use. It is similar to our approach, however, in that (1) interrupts are recognized as a key source of non-determinism and (2) schedule-driven protocol processing is proposed as a solution.

While RIO shares many elements of the approaches described above, we have combined these concepts to create the first vertically integrated real-time ORB endsystem. The resulting ORB endsystem provides scalable performance, periodic processing guarantees and bounded latency, as well as an end-to-end solution for real-time distributed object computing middleware and applications.

4.3 Related Work on High-performance and QoS-enabled Middleware

QoS-enabled middleware is an emerging field of study. Moreover, an increasing number of research efforts are focusing on integrating QoS and real-time scheduling into middleware like CORBA. Our project is unique, however, in that it not only proposes to support end-to-end QoS but to also integrate into the framework support for network signaling protocols, such as UNI 4.0, PNNI, RSVP, and GSMP. The following paragraphs review other approaches to supporting end-to-end QoS.

Xbind: The COMET group at Columbia University has produced a distributed application development environment using CORBA called XBIND [81]. However, our approach is unique in that we are using a real-time, high-performance ORB, *i.e.*, TAO, at the core of our proposed middleware framework. We propose to embed TAO in high-performance network switching elements to support an even more efficient and integrated approach to network element management and control.

BBN QuO: The *Quality Objects* (QuO) distributed object middleware is developed at BBN Technologies [82]. QuO is based on CORBA and provides the following support for agile applications running in wide-area networks: (1) provides *run-time performance tuning and configuration* through the specification of operating regions, behavior alternatives, and reconfiguration strategies that allows the QuO run-time to adaptively trigger reconfiguration as system conditions change (represented by transitions between operating regions), (2) gives *feedback* across software and distribution boundaries based on a control loop in which client applications and server objects request levels of service and are notified of changes in service, and (3) supports *code mobility* that enables QuO to migrate object functionality into local address spaces in order to tune performance and to further support highly optimized adaptive reconfiguration.

UCSB Realize: The Realize project at UCSB [83] supports soft real-time resource management of CORBA distributed systems. Realize aims to reduce the difficulty of developing real-time systems and to permit distributed real-time programs to be programmed, tested, and debugged as easily as single sequential programs. Realize integrates distributed real-time scheduling with fault-tolerance, fault-tolerance with totally-ordered multicasting, and totally-ordered multicasting with distributed real-time scheduling, within the context of OO programming and existing standard operating systems. The Realize resource management model can be hosted on top of TAO [83].

URI TDMI: Wolfe, *et al.*, are developing a real-time CORBA system at the US Navy Research and Development Laboratories (NRaD) and the University of Rhode Island (URI) [84]. The system supports expression and enforcement of dynamic end-to-end timing constraints through timed distributed operation invocations (`TDMIs`) [85]. A `TDMI` corresponds to TAO's `RT_Operation` [6]. Likewise, an `RT_Environment` structure contains QoS parameters similar to those in TAO's `RT_Info`.

One difference between TAO and the URI approaches is that `TDMIs` express required timing constraints, *e.g.*, deadlines relative to the current time, whereas `RT_Operations` publish their resource, *e.g.*, CPU time, requirements. The difference in approaches may reflect the different time scales, seconds versus milliseconds, respectively, and scheduling requirements, dynamic versus static, of the initial application targets. However, the approaches should be equivalent with respect to system schedulability and analysis.

UCI TMO: The Time-triggered Message-triggered Objects (TMO) project [86] at the University of California, Irvine, supports the integrated design of distributed OO systems and real-time simulators of their operating environments. The TMO model provides structured timing semantics for distributed real-time object-oriented applications by extending conventional invocation semantics for object methods (*i.e.*, CORBA operations) to include (1) invocation of time-triggered operations based on system times and (2) invocation and time bounded execution of conventional message-triggered operations.

4.4 Related Work on Open Signaling

Open signaling is another emerging field of study. Below, we review a number of activities on network signaling.

Washington University's CMAP and CMNP: In previous research Washington University defined two ATM signaling protocols, the Connection Management Access Protocol CMAP [87] and the Connection Management Network Protocol CMNP (Draft) [88]. These protocols support a dynamic, general multi-point call model. The call model has (1) separate call and connection control, (2) multiple connections per call, (3) generalized multicast communications, (4) heterogeneous endpoint participation, (5) dynamic addition and deletion of connections and endpoints, and (6) dynamic modification of call, connection and endpoint attributes.

GSMP: Other signaling work at Washington University has focused on prototyping of an OO General Switch management Protocol (GSMP) Version 1.1 [89] implementation. GSMP is a protocol developed by Ipsilon Networks and available as Internet RFC 1987 [89] and RFC 2297 [12]. It is a general-purpose protocol designed to control an ATM switch by allowing a signaling processor to establish and release connections across the switch, to add and delete leaves on a point-to-multipoint connection, to manage switch ports, request configuration information, and to request statistics. The GSMP working group within the IETF is continuing to advance GSMP.

As part of their XBIND effort, the COMET group defined an enhanced version of the IETF's General Switch Management Protocol (GSMP) [12]. qGSMP has been defined in the context of the OPENSIG group and it provides additional messages for requesting QoS from network elements. The IEEE P1520 [90] working group is considering a variant of qGSMP for the Proposed IEEE Standard for Application Programming Interfaces for Networks.

Washington University's Center for Distributed Object Computing is currently building upon earlier signaling work to include new enhanced QoS features defined in GSMP Version 2.0 [12] and qGSMP [91]. In addition, we are tracking the work of the the IETF GSMP working group [92], the OPENSIG groups and the Multiservice Switching Forum (MSF) [93] and the Virtual Switch Interface specification.

SEAN: There are open-source implementations of the ATM signaling protocols which we will review for inclusion into our framework. The ATM Signaling Research Group, Center for Computational Sciences, Naval Research Laboratory has made freely available ATM signaling implementation called SEAN [11]. SEAN includes a host native ATM protocol stack and implements the ATM User Network Interface ITU Q.2931 specification, the ITU Q.2971 extension, and the ATM Forum extension UNI-4.0.

In our proposed effort, we will leverage many of these existing implementations and research results. With the exception of XBIND none of these approaches use standard OO communication middleware as their programming model for signaling. By leverging these existing signaling protocols and implementations and integrating them with CORBA middleware, we can capitalize on the existing software base while presenting NGI application developers with robust, efficient, intuitive, and standard OO programming APIs.

5 Concluding Remarks

Next Generation Internet (NGI) applications will require advanced networks and middleware to support a wide mix of dynamically changing multimedia streams. These streams result from activities like high-bandwidth data acquisition; transparent, across-the-Internet data cache updates; and remote collaboration through interactive, multimedia teleconferencing. The quality, and sometimes the feasibility, of these activities are directly related to the ability of the networks and middleware to provide these applications with their necessary end-to-end quality of service (QoS).

To improve the integration of high-speed networks, I/O subsystems, and higher-level middleware, we propose a three year project that will substantially improve core network and middleware technologies available for NGI applications that run over high-speed LANS and WANs. We will accomplish this by integrating and enhancing three proven technologies – WUGS, TAO, and QuO – that we have developed under prior and ongoing government and industry sponsorship. In our mutually synergistic project, WUGS [7] provides the *intelligent, very high-speed networking infrastructure*, TAO [9] provides an *open source, standards-based, high-performance and predictable middleware*, and QuO [94] provides complementary middleware that supports *adaptivity and end-to-end network, endsystem, and application control*. The result will be the first open-source, standards-based, *vertically* (*i.e.*, network interface $\leftrightarrow$ application layer) and *horizontally* (*i.e.*, end-to-end) integrated high-speed network and middleware infrastructure.

For network users and NGI applications, the benefits of our integrated WUGS/RIO/TAO infrastructure will include: (1) advanced communication capabilities over very high-speed gigabit networks, (2) highly scalable, hardware-supported multicast communication facilities which directly supports the efficient handling of many-to-many and many-to-one multicast without requiring one-to-many overlays, (3) open source, standards-based, high-performance, real-time middleware that is flexible, adaptive, and easy to program and use, (4) new application-oriented capabilities for rapid, dynamic rebinding and resynchronizing collections of cooperating elements, and (5) advanced high-speed network-aware collaborative applications with multi-site coordination features.

For application developers and maintainers, the benefits of our integrated WUGS/RIO/TAO infrastructure will include: (1) an environment for developing portable applications that can take full advantage of the available resources, hardware improvements, and COTS OS/middleware capabilities, (2) easier maintenance and upgrading of these applications to add new capabilities and to support future innovations in hardware and OS/middleware, (3) ease in developing single applications that can run on many different network/endsystem configurations, and in the presence of degraded or failed network/hardware components, (4) ease in upgrading existing applications developed for LANs to use the Internet or the NGI, (5) higher level, component-based and OO applications that are efficient, scalable, and predictable, yet are easier to maintain and are reusable than applications developed using earlier software design paradigms, and (6) separation of the functional (*e.g.*, data processing) capabilities of applications from the performance (*i.e.*, low-latency, high-speed data transport) aspects.

References

[1] ATD, "Advanced Technology Demonstration Network." http://www.atd.net/.

[2] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, 2.2 ed., Feb. 1998.

[3] D. Box, *Essential COM*. Addison-Wesley, Reading, MA, 1997.

[4] A. Wollrath, R. Riggs, and J. Waldo, "A Distributed Object Model for the Java System," *USENIX Computing Systems*, vol. 9, November/December 1996.

[5] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," *IEEE Communications Magazine*, vol. 14, February 1997.

[6] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers," *Computer Communications*, vol. 21, pp. 294–324, Apr. 1998.

[7] W. U. A. R. Laboratory, "Gigabit Networking Technology." http://www.arl.wustl.edu/ jst/gigatech/gigatech.html.

[8] F. Kuhns, D. C. Schmidt, and D. L. Levine, "The Design and Performance of RIO – A Real-time I/O Subsystem for ORB Endsystems," in *Proceedings of the* 5^{th} *IEEE Real-Time Technology and Applications Symposium*, (Vancouver, British Columbia, Canada), IEEE, June 1999.

[9] Center for Distributed Object Computing, "TAO: A High-performance, Real-time Object Request Broker (ORB)." www.cs.wustl.edu/∼schmidt/TAO.html, Washington University.

[10] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," *ICICE Transactions*, 1998.

[11] N. R. L. ATM Signalling Research Group, Center for Computational Sciences, "SEAN: Signalling Entity for ATM Networks ." https://www.nrl.navy.mil/ccs/project/public/sean/SEAN-dev.html.

[12] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall, "Ipsilon's General Switch Management Protocol Specification Version 2.0," Standards Track RFC 2297, Network Working Group, March 1998.

[13] Z. D. Dittia, J. Jerome R. Cox, and G. M. Parulkar, "Design of the APIC: A High Performance ATM Host-Network Interface Chip," in *IEEE INFOCOM '95*, (Boston, USA), pp. 179–187, IEEE Computer Society Press, April 1995.

[14] Z. Dittia and G. Parulkar, "The APIC Approach to High Performance Network Interface Design: Protected DMA and Other Techniques," Tech. Rep. 96-12, Washington University Department of Computer Science, March 1996. submitted for publication.

[15] Z. D. Dittia, G. M. Parulkar, and J. Cox, Jr., "Design and Implementation of a Versatile Multimedia Network Interface and I/O Chip," in *NOSSDAV*, 1996.

[16] F. Kuhns, C. O'Ryan, D. C. Schmidt, and J. Parsons, "The Design and Performance of a Pluggable Protocols Framework for Object Request Broker Middleware," in *Submitted to the IFIP* 6^{th} *International Workshop on Protocols For High-Speed Networks (PfHSN '99)*, (Salem, MA), IFIP, August 1999.

[17] SLAC, "BaBar Collaboration Home Page." http://www.slac.stanford.edu/BFROOT/.

[18] A. Kruse, "CMS Online Event Filtering," in *Computing in High-energy Physics (CHEP 97)*, (Berlin, Germany), Apr. 1997.

[19] T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The Design and Performance of a Real-time CORBA Event Service," in *Proceedings of OOPSLA '97*, (Atlanta, GA), ACM, October 1997.

[20] D. C. Schmidt, "A Family of Design Patterns for Application-level Gateways," *The Theory and Practice of Object Systems (Special Issue on Patterns and Pattern Languages)*, vol. 2, no. 1, 1996.

[21] P. Jain, S. Widoff, and D. C. Schmidt, "The Design and Performance of MedJava – Experience Developing Performance-Sensitive Distributed Applications with Java," *IEE/BCS Distributed Systems Engineering Journal*, 1998.

[22] I. Pyarali, T. H. Harrison, and D. C. Schmidt, "Design and Performance of an Object-Oriented Framework for High-Performance Electronic Medical Imaging," *USENIX Computing Systems*, vol. 9, November/December 1996.

[23] Center for Distributed Object Computing, "Successful Project Deployment of ACE and TAO." www.cs.wustl.edu/∼schmidt/users.html, Washington University.

[24] K. Kavi, J. C. Browne, and A. Tripathi, "Computer Systems Research: The Pressure is On," *IEEE Computer*, vol. 32, pp. 30–39, Jan. 1999.

[25] J. Turner, "An optimal nonblocking multicast virtual circuit switch," in *Proceedings of the Conference on Computer Communications (INFOCOM)*, pp. 298–305, June 1994.

[26] J. S. Turner, "Gigabit Network Kits." http://www.arl.wustl.edu/gigabitkits/kits.html.

[27] I. STS Technologies. http://www.ststech.com/.

[28] Z. D. Dittia, "ATM Port Interconnect Chip." http://www.arl.wustl.edu/apic.html.

[29] Z. D. Dittia, G. M. Parulkar, and J. Jerome R. Cox, "The APIC Approach to High Performance Network Interface Design: Protected DMA and Other Techniques," in *Proceedings of INFOCOM '97*, (Kobe, Japan), IEEE, April 1997.

[30] W. N. Eatherton and T. Aramaki, "SPC Specification," Applied Research Lab, Working Notes ARL-WN-98-02, Washington University, St. Louis, 1998.

[31] D. Decasper, G. Parulkar, S. Choi, J. DeHart, T. Wolf, and B. Plattner, "A Scalable, High Performance Active Network Node," *IEEE Network Magazine*, vol. 13, January/February 1999.

[32] VSI/1.0, "Virtual Switch Interface (VSI) Specification, version 1.0." http://www.msforum.org/switch_control.pdf.

[33] D. C. Schmidt, A. Gokhale, T. Harrison, and G. Parulkar, "A High-Performance Endsystem Architecture for Real-time CORBA," *IEEE Communications Magazine*, vol. 14, February 1997.

[34] R. S. Madukkarumukumana and H. V. Shah and C. Pu, "Harnessing User-Level Networking Architectures for Distributed Object Computing over High-Speed Networks," in *Proceedings of the 2nd Usenix Windows NT Symposium*, August 1998.

[35] T. v. Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net: A User-Level Network Interface for Parallel and Distributed Computing," in *15th ACM Symposium on Operating System Principles*, ACM, December 1995.

[36] P. Druschel and G. Banga, "Lazy Receiver Processing (LRP): A Network Subsystem Architecture for Server Systems," in *Proceedings of the* 1^{st} *Symposium on Operating Systems Design and Implementation*, USENIX Association, October 1996.

[37] L. Krishnamurthy, *AQUA: An Adaptive QUality of Service Architecture for Distributed Multimedia Applications*. PhD thesis, University of Kentucky, 1997.

[38] T. B. Vincent Roca and C. Diot, "Demultiplexed Architectures: A Solution for Efficient STREAMS-Based Communication Stacks," *IEEE Network Magazine*, vol. 7, July 1997.

[39] J. C. Mogul and K. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-driver Kernel," in *Proceedings of the USENIX 1996 Annual Technical Conference*, (San Diego, CA), USENIX, Jan. 1996.

[40] C. Cranor and G. Parulkar, "Universal Continuous Media I/O: Design and Implementation," Tech. Rep. 94-34, Washington University Department of Computer Science, December 1994.

[41] M. Buddhikot, J. Chen, X., D. Wu, and G. Parulkar, "Enhancements to 4.4 BSD UNIX for Networked Multimedia in Project MARS," in *Proceedings IEEE Multimedia Systems'98*, June 1998.

[42] R. Gopalakrishnan and G. Parulkar, "Quality of Service Support for Protocol Processing Within Endsystems," in *High-Speed Networking for Multimedia Applications* (W. Effelsberg, *et. al.*, ed.), Kluwer Academic Publishers, 1995.

[43] R. Gopalakrishnan and G. M. Parulkar, "Efficient User Space Protocol Implementations with QoS Guarantees using Real-time Upcalls," Tech. Rep. 96-11, Washington University Department of Computer Science, March 1996.

[44] C. Cranor and G. Parulkar, "Design of Universal Continuous Media I/O," in *Proceedings of the 5th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '95)*, (Durham, New Hampshire), pp. 83–86, Apr. 1995.

[45] R. Gopalakrishnan and G. Parulkar, "A Real-time Upcall Facility for Protocol Processing with QoS Guarantees," in 15^{th} *Symposium on Operating System Principles (poster session)*, (Copper Mountain Resort, Boulder, CO), ACM, Dec. 1995.

[46] R. Gopalakrishnan and G. M. Parulkar, "Efficient Quality of Service Support in Multimedia Computer Operating Systems," Tech. Rep. 94-26, Dept. of Computer Science, Washington University in St. Louis, 1994.

[47] M. Buddhikot and G. Parulkar, "Scalable Multimedia-On-Demand via World-Wide-Web (WWW) with QOS Guarantees," in *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio, Video (NOSSDAV)*, 1996.

[48] R. Gopalakrishnan and G. M. Parulkar, "RMDP-A Real-time CPU Scheduling Algorithm to Provide Guarantees for Protocol Processing," in *IEEE Real-time Technology and Applications Symposium, (Poster)*, May 1995.

[49] M. Buddhikot and G. Parulkar, "Efficient Data Layout, Scheduling and Playout Control in MARS," in *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio, Video (NOSSDAV)*, April 1995.

[50] I. Pyarali, C. O'Ryan, D. C. Schmidt, N. Wang, V. Kachroo, and A. Gokhale, "Applying Optimization Patterns to the Design of Real-time ORBs," in *Proceedings of the* 5^{th} *Conference on Object-Oriented Technologies and Systems*, (San Diego, CA), USENIX, May 1999.

[51] Compaq, Intel, and Microsoft, "Virtual Interface Architecture, Version 1.0." http://www.viarch.org, 1997.

[52] R. Rajkumar, L. Sha, and J. P. Lehoczky, "Real-Time Synchronization Protocols for Multiprocessors," in *Proceedings of the Real-Time Systems Symposium*, (Huntsville, Alabama), December 1988.

[53] S. Khanna and et. al., "Realtime Scheduling in SunOS 5.0," in *Proceedings of the USENIX Winter Conference*, pp. 375–390, USENIX Association, 1992.

[54] A. Gokhale and D. C. Schmidt, "Measuring the Performance of Communication Middleware on High-Speed Networks," in *Proceedings of SIGCOMM '96*, (Stanford, CA), pp. 306–317, ACM, August 1996.

[55] A. Gokhale and D. C. Schmidt, "Measuring and Optimizing CORBA Latency and Scalability Over High-speed Networks," *Transactions on Computing*, vol. 47, no. 4, 1998.

[56] E. Eide, K. Frei, B. Ford, J. Lepreau, and G. Lindstrom, "Flick: A Flexible, Optimizing IDL Compiler," in *Proceedings of ACM SIGPLAN '97 Conference on Programming Language Design and Implementation (PLDI)*, (Las Vegas, NV), ACM, June 1997.

[57] A. Gokhale and D. C. Schmidt, "Optimizing a CORBA IIOP Protocol Engine for Minimal Footprint Multimedia Systems," *Journal on Selected Areas in Communications special issue on Service Enabling Platforms for Networked Multimedia Systems*, to appear, 1999.

[58] A. Gokhale and D. C. Schmidt, "The Performance of the CORBA Dynamic Invocation Interface and Dynamic Skeleton Interface over High-Speed ATM Networks," in *Proceedings of GLOBECOM '96*, (London, England), pp. 50–56, IEEE, November 1996.

[59] A. Gokhale and D. C. Schmidt, "Evaluating the Performance of Demultiplexing Strategies for Real-time CORBA," in *Proceedings of GLOBECOM '97*, (Phoenix, AZ), IEEE, November 1997.

[60] T. A. Forum, "The ATM Forum Technical Specifications." http://www.atmforum.com/atmforum/specs/specs.html.

[61] R. Braden et al, "Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification." Internet Draft, May 1997. ftp://ietf.org/internet-drafts/draft-ietf-rsvp-spec-15.txt.

[62] G. Parulkar, D. C. Schmidt, and J. S. Turner, "$\mathrm{a^It^Pm}$: a Strategy for Integrating IP with ATM," in *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM)*, ACM, September 1995.

[63] D. C. Schmidt, S. Mungee, S. Flores-Gaitan, and A. Gokhale, "Software Architectures for Reducing Priority Inversion and Non-determinism in Real-time Object Request Brokers," *Journal of Real-time Systems*, To appear 1999.

[64] C. D. Gill, D. L. Levine, and D. C. Schmidt, "Evaluating Strategies for Real-Time CORBA Dynamic Scheduling," *The International Journal of Time-Critical Computing Systems, special issue on Real-Time Middleware*, 1999, to appear.

[65] D. C. Schmidt and T. Suda, "An Object-Oriented Framework for Dynamically Configuring Extensible Distributed Communication Systems," *IEE/BCS Distributed Systems Engineering Journal (Special Issue on Configurable Distributed Systems)*, vol. 2, pp. 280–293, December 1994.

[66] Object Management Group, *Minimum CORBA - Joint Revised Submission*, OMG Document orbos/98-08-04 ed., August 1998.

[67] M. Waldvogel, G. Varghese, J. S. Turner, and B. Plattner, "Scalable High Speed IP Routing Lookups," in *Proceedings of SIGCOMM '97*, (Cannes, France), ACM, August 1997.

[68] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner, "Router plugins - a modular and extensible software framework for modern high performance integrated services routers," Department of Computer Science TR-98-08, Washington University in St. Louis, February 1998.

[69] H. Andiseshu and G. Parulkar, "A State Management Protocol for IntServ, DiffServ and Label Switching," in *Proceedings of the 1997 International Conference on Network Protocols (ICNP 98)*, IEEE Computer Society, 1998.

[70] H. Adiseshu, G. Parulkar, and S. Suri, "A Simplified Reservation and State Setup Protocol," Department of Computer Science, Technical Report WUCS-98-07, Washington University, St. Louis, 1998.

[71] S. Mungee, N. Surendran, and D. C. Schmidt, "The Design and Performance of a CORBA Audio/Video Streaming Service," in *Proceedings of the Hawaiian International Conference on System Sciences*, Jan. 1999.

[72] G. Parulkar, D. C. Schmidt, E. Kraemer, J. Turner, and A. Kantawala, "An Architecture for Monitoring, Visualization, and Control and Gigabit Networks," *IEEE Network*, vol. 11, September/October 1997.

[73] MCI and NSF, "Very high performance Backbone Network Service (vBNS)." http://www.vbns.net/.

[74] CAIRN, "Collaborative Advanced Internet Research Network (CAIRN)." http://www.cairn.net/.

[75] DARPA, NASA, and NSF, "The National Transparent Optical Network (NTON)." http://www.ntonc.net/.

[76] J. Mogul and S. Deering, "Path MTU Discovery," *Network Information Center RFC 1191*, pp. 1–19, Apr. 1990.

[77] A. B. Montz, D. Mosberger, S. W. O'Malley, L. L. Peterson, T. A. P. sting, and J. H. Hartman, "Scout: A communications-oriented operating system," Tech. Rep. 94-20, Department of Computer Science, University of Arizona, June 1994.

[78] D. Mosberger and L. Peterson, "Making Paths Explicit in the Scout Operating System," in *Proceedings of OSDI '96*, Oct. 1996.

[79] B. Bershad, "Extensibility, Safety, and Performance in the Spin Operating System," in *Proceedings of the 15^{th} ACM SOSP*, pp. 267–284, 1995.

[80] M. Fiuczynski and B. Bershad, "An Extensible Protocol Architecture for Application-Specific Networking," in *Proceedings of the 1996 Winter USENIX Conference*, Jan. 1996.

[81] T. C. Group, "XBIND." http://comet.ctr.columbia.edu/.

[82] J. A. Zinky, D. E. Bakken, and R. Schantz, "Architectural Support for Quality of Service for CORBA Objects," *Theory and Practice of Object Systems*, vol. 3, no. 1, 1997.

[83] V. Kalogeraki, P. Melliar-Smith, and L. Moser, "Soft Real-Time Resource Management in CORBA Distributed Systems," in *Proceedings of the Workshop on Middleware for Real-Time Systems and Services*, (San Francisco, CA), IEEE, December 1997.

[84] V. F. Wolfe, L. C. DiPippo, R. Ginis, M. Squadrito, S. Wohlever, I. Zykh, and R. Johnston, "Real-Time CORBA," in *Proceedings of the Third IEEE Real-Time Technology and Applications Symposium*, (Montréal, Canada), June 1997.

[85] V. Fay-Wolfe, J. K. Black, B. Thuraisingham, and P. Krupp, "Real-time Method Invocations in Distributed Environments," Tech. Rep. 95-244, University of Rhode Island, Department of Computer Science and Statistics, 1995.

[86] K. H. K. Kim, "Object Structures for Real-Time Systems and Simulators," *IEEE Computer*, pp. 62–70, Aug. 1997.

[87] K. Cox and J. DeHart, "Connection Management Access Protocol Specification (CMAP), Version 3.0," Department of Computer Science, Technical Report WUCS-94-21, Washington University, St. Louis, July 1994.

[88] J. DeHart and D. Wu, *Connection Management Network Protocol Specification (CMNP)*, Version 1.0 Draft ed., July 1994.

[89] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall, "Ipsilon's General Switch Management Protocol Specification Version 1.1," Standards Track RFC 1987, Network Working Group, August 1996.

[90] IEEE, "IEEE P1520, Proposed IEEE Standard for Application Programming Interfaces for Networks." http://www.ieee-pin.org/.

[91] Constantin M. Adam, Aurel A. Lazar, and Mahesan Nandikesan, *Draft Technology Submission Working Document, Programming Interfaces for Networks*. Princeton, January 1999.

[92] P. Ranganathan, D. Sreenivasamurthy, J. Evans, and A. Kaushal, "A framework for QoS support for open control," Tech. Rep. Draft RFC, IETF GSMP Working Group, Pittsburgh, PA, December 1998.

[93] MSF, "Multiservice Switching Forum." http://www.msforum.org/.

[94] B. Technologies, "Quality Objects (QuO)." http://www.dist-systems.bbn.com/papers.

[95] R. Lachenmaier, "Open Systems Architecture Puts Six Bombs on Target." www.cs.wustl.edu/∼schmidt/TAO-boeing.html, Dec. 1998.

[96] J. Hu, I. Pyarali, and D. C. Schmidt, "Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks," in *Proceedings of the 2^{nd} Global Internet Conference*, IEEE, November 1997.

[97] J. Hu, S. Mungee, and D. C. Schmidt, "Principles for Developing and Measuring High-performance Web Servers over ATM," in *Proceeedings of INFOCOM '98*, March/April 1998.

[98] J. O. Coplien and D. C. Schmidt, eds., *Pattern Languages of Program Design*. Reading, MA: Addison-Wesley, 1995.

[99] M. Fayad, R. Johnson, and D. C. Schmidt, eds., *Object-Oriented Application Frameworks: Problems & Perspectives*. New York, NY: Wiley & Sons, 1999.

[100] M. Fayad, R. Johnson, and D. C. Schmidt, eds., *Object-Oriented Application Frameworks: Applications & Experiences*. New York, NY: Wiley & Sons, 1999.

A Facilities

Washington University has excellent facilities and infrastructure to support the proposed research program. Computing facilities in the department include over 200 servers and workstations for use by faculty, research staff and students in various labs and centers. All these reside on the University's campus-wide network, which provides communication with the campus and the Internet. Washington University also has one of the largest operational campus ATM networks.

The proposed research project will be carried out in Washington University's Applied Research Laboratory (ARL) and Center for Distributed Object Computing (DOC). The purpose of ARL and DOC is to develop high performance hardware and real-time software/hardware technologies by building practical prototype systems and deploying them in testbed and production settings [95].

Both ARL and the DOC Center have several projects in progress, with Projects Zeus and TAO being the oldest. The goal of Project Zeus has been to develop an ATM campus network that can support multirate data, CD quality audio, video, and high resolution images in an integrated fashion. The phase-0 of this project led to prototyping of the ATM switch and then to the nation's first multi-switch metropolitan area ATM network. The goal of the TAO project is to develop flexible, efficient, and predictable OO communication middleware based on CORBA that can automate common communication software tasks, such as connection establishment, event demultiplexing, event handler dispatching, message routing, dynamic configuration of services, priority-based real-time thread scheduling, and flexible management of parallel protocol and service processing.

Our existing network includes eleven 16-port, 155 Mb/s per port (commercial versions of our own) ATM switches and over 50 endpoints that include multimedia workstations, compute and storage servers, and other interesting imaging devices. Connected by several hundred miles of fiber optic cable, these switches are located throughout the Hilltop and Medical campuses and also provide a connection to Barnes West County Hospital. The network supports routine network applications and a variety of multimedia and imaging applications including multi participant teleconferencing and collaboration, electronic radiology, and video-on-demand.

The multimedia capability at a workstation is provided by a platform-independent "pizza box" called the MultiMedia eXplorer (MMX). The MMX incorporates an ATM network adapter, an ATM extension port, an RS232 link for control by the workstation, a full-duplex motion JPEG codec for production quality video, a DSP-based codec for CD quality audio, and a high-speed serial port for the delivery of high-resolution images.

ARL's Washington University Gigabit Switching (WUGS) ATM Testbed Project (sponsored by ARPA/ITO) is concerned with the design of two key gigabit network technologies: highly scalable multicast ATM switching systems and host-network interfaces suitable for supporting gigabit data rates to and from application-level programs. Delivering gigabits to applications has meant rethinking the host I/O subsystem and protocols and their implementations within the host operating system. The project will culminate in the creation of a gigabit testbed with multiple switching systems supporting link speeds of 600 Mb/s, 1.2 Gb/s and 2.4 Gb/s and supporting workstation and server applications at 1.2 Gb/s.

The DOC Center's TAO project (sponsored by ARPA/ITO, NSF, and many major telecommunication and aerospace companies) provides an open-source, standards-based, high-performance, real-time ORB endsystem that supports applications with deterministic and statistical QoS requirements, as well as "best-effort" requirements, and is the first ORB to support end-to-end QoS guarantees over ATM/IP networks [62]. TAO's features and optimizations include an ORB Core that minimizes context switching, synchronization, dynamic memory allocation, and data movement [63]; a highly-scalable Object Adapter that demultiplexes requests in constant-time [50]; an optimizing IDL compiler [57]; real-time I/O subsystem [8], and a global resource allocation and scheduling framework [6].

ARL and and DOC currently have a combined full-time hardware and software engineering staff of 15 individuals who provide the continuity and consistency required to ensure that demanding systems projects, such as this one, can be carried through to a predictable completion.

B Technology Transfer Plan

An important goal of the Applied Research Laboratory (ARL) and the Center for Distributed Object Computing (DOC) is to develop high performance hardware and software technologies by building practical prototype systems and deploying them in production and testbed settings. Research ideas, particularly in the area of computing and communication, increasingly require the construction of prototypes and a capability to produce timely demonstrations. Industry collaborators find it essential to understand performance and engineering issues before making large commitments to new products. Moreover, the rapid development of prototypes in universities can often substantially reduce the time-to-market for new products.

ARL and the DOC Center take pride in prototyping systems that can be licensed to industry for product development and external R&D. ARL has licensed the following technologies from its laboratory:

- A scalable ATM switch architecture and associated ASIC (Application Specific Integrated Circuit) designs to SynOptics, a Santa Clara, California-based company with offices worldwide (SynOptics later became Bay Networks and was recently acquired by Nortel).
- ATM signaling software to SynOptics, Southwestern Bell Co., and Ascom Nexion, a Massachusetts based company with a software division in St. Louis (Ascom Nexion was recently acquired by Fujitsu).
- MMX, the multimedia explorer, a hardware and software system which allows any workstation to do high performance multimedia over an ATM network, to STS Technologies, a St. Louis-based spinoff company.
- Gigabit ATM switches and associated network interface cards. This technology has been distributed to 30 universities in the form of *Gigabit Network Kits*, providing a flexible, open research platform that can be modified and extended to pursue a variety of research objectives. This technology has been licensed to STS Technologies.
- Advanced network technology for the construction of high performance routing switches capable of routing IP packets at link rates of 10 Gb/s and with aggregate system capacities of move than 10 Tb/s. This technology has been licensed to Growth Networks, Inc., a California-based startup company.

Likewise, the DOC Center has developed the following middleware technologies, which are now supported commercially and used in many industry R&D centers and product groups:

- The ADAPTIVE Communication Environment (ACE) – ACE is a widely used OO framework containing components that implement key patterns for high-performance and real-time communication systems [65]. ACE has been used for many communication software systems in research labs and commercial projects, including Bellcore, Boeing, CERN, Cisco, DEC, Ericsson, Kodak, JPL, Lucent, Lockheed Martin, Motorola, NASA, Raytheon, SAIC, Siemens, SLAC, and StorTek. A commercial company, Riverace, supports ACE using an open-source software model.
- The ACE ORB (TAO) – TAO is a high-performance, real-time ORB endsystem targeted for applications with deterministic and statistical QoS requirements, as well as best effort requirements [6]. The TAO ORB endsystem is developed using components in ACE and has been used in many commercial products, including a family of avionics mission computing systems at Boeing [19], satellite communications at Raytheon, and HLA RTI simulation middleware at SAIC. A commercial company, OCI, supports TAO using an open-source software model.
- The JAWS Adaptive Web Server (JAWS) – JAWS is a high-performance, adaptive Web server framework [96, 97]. It is also developed using components in ACE and has been integrated into commercial systems at a number of companies, including Siemens. A commercial company, Entera, supports JAWS using an open-source software model.

C Biographical Sketches and Qualifications

The Co-PIs for the project are Douglas Schmidt, Jonathan Turner, Fred Kuhns, and David Levine from Washington University. As indicated below, the team of researchers is uniquely qualified to undertake the proposed effort. In particular, Dr. Schmidt is a leading expert in the area of distributed object computing systems and real-time middleware. Dr. Turner is a leading expert in the field in the design, implementation, and deployment of high speed ATM networks. Furthermore, our team has extensive experience building large-scale distributed applications, middleware, and high-speed networks and transferring their state-of-the-art technology to government R&D groups, universities, and commercial ventures.

Dr. Douglas C. Schmidt is an Associate Professor and Director of the Center for Distributed Object Computing in the Department of Computer Science and in the Department of Radiology at Washington University in St. Louis, Missouri, USA. His research focuses on design patterns, implementation, and experimental analysis of object-oriented techniques that facilitate the development of high-performance, real-time distributed object computing systems on parallel processing platforms running over high-speed ATM networks and embedded system interconnects.

Dr. Schmidt is an internationally recognized expert on distributed object computing and real-time CORBA middleware, and has published over 75 papers in IEEE, ACM, IFIP, and USENIX journals, technical conferences, and workshops. He is the chief architect and designer for two significant software systems: (1) The ADAPTIVE Communication Environment (ACE), which is a widely used OO framework containing components that implement key patterns for high-performance and real-time communication systems [65] and (2) the ACE ORB (TAO) which is a high-performance, real-time ORB [6, 19].

Dr. Schmidt is the editor of several books on patterns [98] and frameworks [99, 100] and is currently writing two book on these topics for Addison Wesley and Wiley & Sons. In addition to his academic research, Dr. Schmidt has served as a consultant for dozens of companies, including Ericsson, Motorola Iridium, Siemens, Kodak, Boeing, Lucent, Nortel, and Lockheed, where he has helped develop real-time avionics computing systems, distributed telecommunication switch management systems, network management software for global personal communications systems, and high performance distributed medical imaging systems over ATM networks. He is a member of IEEE, ACM, and USENIX.

Dr. Jonathan S. Turner is Sever Professor of Engineering, and Director of the Applied Research Laboratory at Washington University. He is also former chairman of the Department of Computer Science and a co-founder of Growth Networks Inc., a venture-funded startup company that designs, develops and markets scalable network and communication products for the WAN market. He received the MS and PhD degrees in computer science from Northwestern University in 1979 and 1981. He is an internationally recognized expert in the design and analysis of switching systems, especially with respect to multicast ATM networks. He has been awarded more than a dozen patents for his work on switching systems and has many widely cited publications in this area.

Dr. Turner has been engaged in research in high speed networks for more than fifteen years. At Bell Laboratories, in the early eighties, he was the lead system architect for a project that sought to integrate voice and data communication using high performance hardware-based packet switching. This *fast packet switching* technology stimulated the development of frame relay and ATM. At Washington University he led a project leading to one of the first scalable multicast switch architectures for ATM. This technology was later licensed to SynOptics, which used it in their first commercial ATM switch products. More recently, he has led the development of switching system technology that can support link speeds of 2.4 Gb/s and aggregate system capacities of over one terabit per second. This technology has now been distributed in the form of *Gigabit Network Kits* to 30 different universities, providing a flexible, open research platform that can be freely modified and extended to pursue a wide variety of research agendas.

Dr. Turner's research interests also include the study of algorithms and computational complexity, with particular interest in the probable performance of heuristic algorithms for NP-complete problems. He is a member of the ACM, SIAM and a fellow of the IEEE.

Fred Kuhns is a Senior Research Associate in the Center for Distributed Object Computing of the Department of Computer Science at Washington University, St. Louis. He received the M.S.E.E. from Washington University, St. Louis, and the B.S.E.E. from the University of Memphis, Memphis TN. His research interests focus on operating system and network support for high-performance, real-time distributed object computing systems. His recent research projects have focused on the design and implementation of real-time I/O subsystems, software support for high-performance interfaces, and QoS support in integrated service routers.

Dr. Levine is a Senior Research Associate in the Center for Distributed Object Computing of the Department of Computer Science at Washington University, St. Louis. He received the Ph.D. in Computer Science from the University of California, Irvine, the M.S.E.E./C.S. from George Washington University, and the B.S.M.E. from Cornell University. His current research interests include testing and performance analysis of real-time systems, and scheduling of distributed real-time systems. In addition, Dr. Levine has contributed substantial amounts to the ADAPTIVE Communication Environment (ACE) framework and The ACE ORB (TAO). Dr. Levine has extensive industry experience developing software for broadband telecommunications, high-fidelity electro-optic sensor system simulation, and both electric/hybrid and internal combustion engine vehicle applications. He is a Registered Professional Engineer in the District of Columbia.