CISCO PROPOSAL

Designing and Optimizing a Real-Time Embedded ORB for Network Element Management and Control Agents

Points of Contact:

Technical Matters:
Douglas C. Schmidt
Associate Professor & Director of the Center for Distributed Object Computing
Department of Computer Science
Washington University in St. Louis
TEL (314) 935-4215
FAX (314) 935-7302
EMAIL schmidt@cs.wustl.edu

Administrative Matters:
Fred Kuhns
Senior Research Associate
Department of Computer Science,
Washington University in St. Louis
TEL (314) 935-6598
FAX (314) 935-7302
EMAIL fredk@cs.wustl.edu

A Statement of Work

The scope of this effort is to create a real-time ORB that's compliant with the new OMG Minimum CORBA [12] specification and embed this ORB within Cisco network elements, such as IP routers and ATM switches. This embedded ORB will leverage and enhance the advanced real-time features [21, 19] developed by Washington University's Center for Distributed Object Computing to implement network element management and control protocols efficiently, scalably, and predictably using standard-based middleware. This proposal describes the specific tasks to be performed during the 12 months of the proposed project.

A.1 Background

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 is commonplace with Gigabit Ethernet and high-speed IP routers becoming more common. In networks and GigaPoPs being deployed for the Next Generation Internet (NGI), such as the Advanced Technology Demonstration Network (ATDnet) [1], SONET interfaces operating at 2.4 Gbps (OC-48) link speeds are deployed while the industry is looking to 10 Gbps links. Future service providers may also deploy 10 Gbps Ethernet and IP. However, the general lack of robust and flexible tools and middleware for provisioning, and controlling these network elements has limited the rate at which NGI applications have been developed to leverage advances in high-speed networks.

What is required is a high-performance, real-time and QoS aware communications middleware embedded in network elements, such as IP routers, as well as ATM and Gigabit Ethernet switches. This embedded ORB middleware can provide a uniform access interface to network management plane and control plane activities. Below, we outline the key components in such a middleware-oriented solution.

The ACE ORB (TAO): The TAO CORBA 2.3-compliant Object Request Broker (ORB) is being developed at Washington University's Center for Distributed Object Computing (DOC) [3]. TAO is 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. [18]. TAO's features and optimizations include an ORB Core that minimizes context switching, synchronization, dynamic memory allocation, and data movement [22]; a highly-scalable Object Adapter that demultiplexes requests in constant-time [19]; an optimizing IDL compiler [6]; real-time I/O subsystem [9], and a global resource allocation and scheduling framework [21, 5].

CORBA protocol model synopsis: CORBA Inter-ORB Protocols (IOP)s define interoperability between ORB endsystems. IOPs provide data representation formats and ORB messaging protocol specifications that can be mapped onto standard and/or customized transport protocols. Regardless of the choice of ORB messaging or transport protocol, a standard programming model is exposed to the CORBA applications. Figure 1 shows the relationships between these various components and layers.

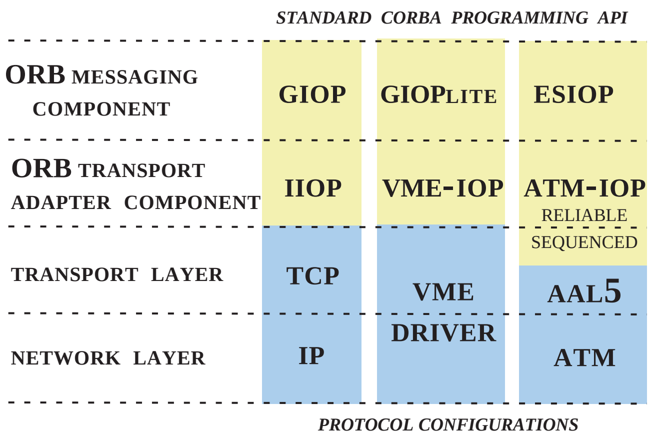

Figure 1: Relationship Between CORBA Programming APIs, Inter-ORB Protocols, and Transport-specific Mappings

A standard General Inter-ORB Protocol (GIOP) is defined by the CORBA protocol interoperability architecture specification [14]. The CORBA specification also defines a transport-specific mapping of GIOP onto the TCP/IP protocol suite. This mapping is called the Internet Inter-ORB Protocol (IIOP) and is required for an ORB implementation to be considered "interoperability compliant." Other mappings of GIOP onto different transport protocols are allowed by the specification. In addition, the standard allows entirely different inter-ORB protocols, known as Environment Specific Inter-ORB Protocols (ESIOP)s, to be configured beneath the standard CORBA programming APIs.

Regardless of whether GIOP or an ESIOP is used, a CORBA IOP must define a data representation, an ORB message format, an ORB transport protocol or transport protocol adapter, and an object addressing format [15].

Pluggable protocol framework: Within the scope of the CORBA interoperability architecture, ORB developers are free to optimize internal data structures and algorithms [19]. Moreover, ORBs may use specialized inter-ORB protocols (ESIOPs) and ORB services and still comply with the specification.[1]

We have leveraged this aspect of the standard and developed a *pluggable protocol framework* within TAO. A key feature

[1]An ORB *must* implement GIOP/IIOP, however, to be interoperability-compliant.

of this framework's design is its decoupling of ORB messaging and transport interfaces from its transport-specific protocol components. Figure 2 shows the partitioning of responsibilities for pluggable protocols and how it relates to other inter-

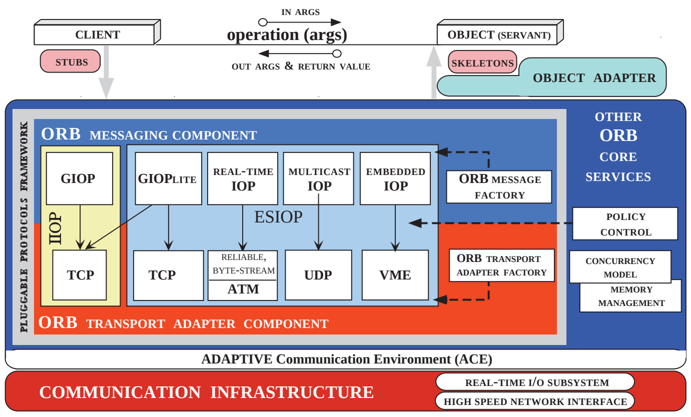

Figure 2: TAO's Pluggable Protocols Framework Architecture

nal ORB components and services. This new framework is transparent to application developers because protocols can be (re)configured without modifying the standard CORBA programming API.

TAO's pluggable protocols framework design allows custom ORB messaging and transport protocols to be configured flexibly and used transparently by CORBA applications. For example, if ORBs communicate over a high-speed networking protocol with QoS support, such as ATM [4] or RSVP [20], then simpler, optimized ORB messaging and transport protocols can be configured to eliminate unnecessary features and overhead of the standard CORBA GIOP and IIOP protocols. Likewise, TAO's pluggable protocols framework makes it straightforward to support customized embedded system interconnects, such as CompactPCI or VMEBus, under the standard CORBA GIOP protocol.

TAO's pluggable protocols framework also supports the creation of efficient, high performance inter-ORB *in-line bridges*. An in-line bridge converts inter-ORB messages or requests from one type of Inter-ORB protocol to another. This feature makes it possible to efficiently bridge disparate ORB domains without incurring unnecessary context switching, synchronization, or data movement. An interesting side benefit of this feature is the ability to "plugin" new ORB messaging protocols, such as the Simple Network Management Protocol [2] (SNMP), Virtual Switch Interface (VSI) [23], or the General Switch Management Protocol (GSMP) [17, 16], which are widely used to manage and control network elements.

MIB and agents: A *MIB* is a logical store of information that is controlled and managed by an *agent*. Agents and MIBs export information on *managed objects*, which contain attributes of network elements that must be tracked to ensure the health of a system. Standard MIBs, such as SNMP and CMIP

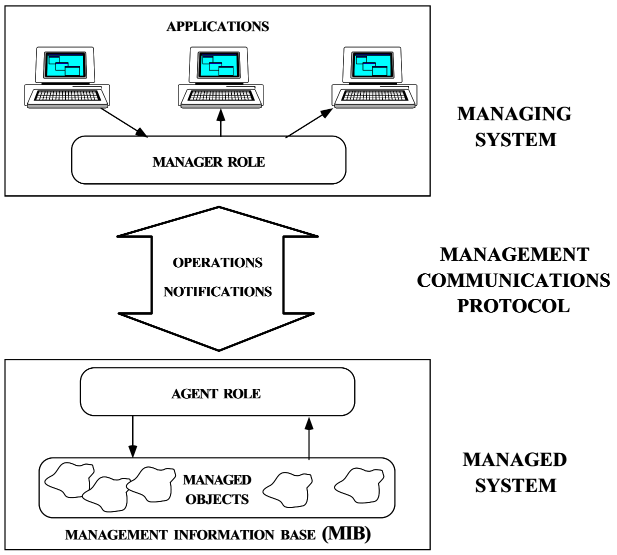

Figure 3: Architecture of Common Network Management Components

MIBs, contain pre-defined objects corresponding to core resources in an Internet environment, *e.g.*, IP, TCP, UDP, network interfaces, etc. In network management for traditional wireline systems, for instance, managed objects maintain information about entities like protocol stacks. Likewise, in a wireless system, managed objects include other information, such as the status of power supplies and antennas, the number of transceivers, and the current allocation of CDMA channels.

Figure 3 illustrates the structure and relationships among the primary "logical" components in the IETF Internet network management reference model. By using a standard schema definition language, such as the SMI standard defined by the IETF, it is possible to define MIBs for use by a variety of wireline and wireless networking systems. However, existing network management and control protocols are tedious and error-prone to use because they do not have standard programming APIs. Moreover, they do not leverage the intense innovation cycles that exist within middleware technologies emerging in the commercial marketplace.

A.2 Proposal: Design and Optimize a Real-Time Embedded ORB for Network Element Management and Control

In our proposed effort we will combine our knowledge of Internet and management and control protocols with our expertise in ORB middleware technologies to conduct a 12 month research program to enhance TAO so that it (1) conforms to the Real-time CORBA [13], Minimum CORBA [12], and Fault Tolerance [11] standards and (2) can be embedded in a Cisco network elements to improve the management and control of applications and services across a wide range of networks. In particular, we will add features and optimizations to create an embedded TAO ORB. This ORB will be used to develop a highly scalable and predictable network management and control agent (NMCA) framework that uses standard CORBA programming APIs and protocols to manage and control network elements without having to develop proxy agents that bridge, *e.g.*, SNMP ↔ IIOP.

The result of this NMCA framework effort will be middleware-based agent framework based on CORBA that will manage and control resources within Cisco network elements, such as high-speed IP routers. Within our framework we will have ORBs (1) embedded on the network elements and (2) resident on the client hosts, the network elements (*e.g.*, router/switch controllers), and agents. The broader goal of our work is to provide a standards-based framework that can provision and monitor end-to-end QoS guarantees for real-time and high-bandwidth applications running over high-speed networks.

To provide a baseline architecture for this work, we will enhance TAO to create a network element embedded CORBA ORB configuration to provide a uniform access interface to network controls, services, and management resources. The embedded ORB configuration will support standard network element management and control operations and interfaces, such as those specified by SNMP and VSI. Unlike conventional middleware efforts, however, the embedded ORB configuration using TAO will be a minimal footprint [12] and real-time [13] version of CORBA that can be installed into Cisco network elements based on ENA. This configuration will improve the flexibility and control of applications and services across high-speed networks without sacrificing performance.

Figure 4 shows a prototypical embedded configuration where multiple management agents and signaling processors communicate with a network element. In this case, all control and management communication occurs within the context of an ORB. An application running on an endstation will request that a connection be established from between itself and another endstation. The signaling processors will process the request, determine a route, and request each switch along the route to allocate the necessary resources. The master controller communicates with a slave controller using the exported VSI-based interfaces. Likewise, an ORB can be used to communication performance management, fault management, and configuration management information between the network elements and a network operations center or management station.

By embedding the ORB in the network element and exposing standard CORBA-based interfaces developers can take advantage of the flexibility and powerful features of CORBA. There is the immediate advantage of simplifying the provisioning, monitoring, and control of network management and control applications. Plus, the applications and services developed using this ORB middleware will yield more modular, extensible, and standard solutions that can be reused across multiple projects and application families. For example, new management and control functionality or interfaces can be added without exposing application developers to underlying details or complexities. Likewise, we can add support for new signaling standards or enhanced features while maintaining a consistent interface to developers of management plane and control plane applications.

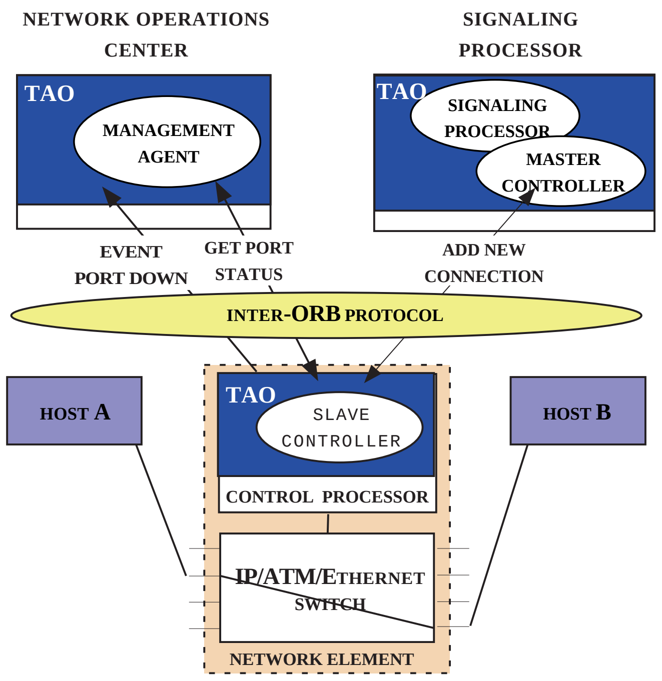

Figure 4: Using the Embedded ORB Configuration for VSI Signaling and Control

The following subsections describe the specific tasks we will perform during the 12 months of the proposed project.

Task 1: Create an OO Network Management and Control Model

Activities: In this task, we will build on network control and management work conducted earlier by Washington University to create an OO network management and control agent (NMCA) model. The specific steps involved in this task include the following:

Step 1 – Develop an object model: We will survey existing SNMP and VSI-based implementations and simulators that manage and control network elements. Using this information, we will then create a CORBA-based object model. This step is primarily concerned with defining CORBA IDL interfaces for network management and control agents.

Step 2 – Map the object model onto a CORBA servant architecture: In this step, we will determine an appropriate mapping from (1) the IDL interfaces defined in step 1 to (2) a set of servants that implement these interfaces. Since one of our goals is to support highly scalable network management and control agents, this step is necessary to evaluate how to apply important CORBA features, such as POA *servant managers* [7], to implement servants for the control and management interfaces.

Deliverables: The deliverables for Task 1 will include the IDL interface definitions and the servant architecture for the NCMA framework, along with a report documenting what we have formulated. This report will identify and describe the key components, policies, and mechanisms related to the OO definition of network management and control using CORBA. The NCMA framework deliverables in Task 1 will provide the basis for subsequent tasks described below.

Task 2 – Enhance the TAO High-Performance, Real-time ORB Middleware

Activities: This task will focus on enhancing the existing TAO ORB to (1) include support for generic QoS specification and QoS enforcement for specific network management and control protocols and (2) produce a minimal footprint, fault tolerant ORB that can be embedded in network elements, as described in the following steps:

Step 1: Define IDL interfaces that map application-level QoS requirements to underlying network and platform mechanisms, using pluggable protocols [8] as an enabling technology.

Step 2: Make TAO comply to the Minimum CORBA specification [12] and embed it in the network elements as an agent, as well as on client hosts and the signaling processors to provide an open network element management and control framework for high-speed networks, such as ATM, Gigabit Ethernet, and high-speed IP routers.

Step 3: Implement key components in the forthcoming CORBA Fault Tolerance specification [11], which insults client applications from managing redundant copies, failure masking, and recovery. This fault tolerance functionality will be supported via (1) multi-profile IORs, (2) passive replication, e.g., via checkpointing and the new OMG Persistent State Service, and (3) active replication, *e.g.*, via ordered multi-cast, such as Totem and Eternal [10].

Deliverables: The deliverables for Task 2 will include the QoS-enabled ORB interfaces and an embeddable version of TAO that conforms to the Minimum CORBA and CORBA Fault Tolerance specifications.

Task 3: Port TAO to a Cisco Network Element and Implement an Optimal IOP

Activities: In this task, we will embed TAO in a Cisco network element and integrate the features from Task 2 using TAO's *pluggable protocols framework* [8]. The specific steps involved in this task include the following:

Step 1 – Network element evaluation: This step involves the following three activities:

1. Acquire Cisco network element hardware/software and verify operation in our environment.
2. Evaluate the configurability of the network element and develop mappings from network management and control messages to element-specific control/management commands or processing requirements.
3. Determine available computing resources, *e.g.*, memory footprint and processor cycles, that are necessary to support the ORB and management/control objects in the network element.

Step 2 – Port TAO to the network element: Evaluate both the network element and necessary ORB functionality to determine which TAO and ACE subsets are required. Based on this analysis, build a minimal TAO with only the components required for the Cisco environment, *e.g.*, QNX/Neutrino and ENA. As part of this step, we will port ACE+TAO to the Greenhills Embedded C++ (EC++) compiler. EC++ will significantly reduce the memory footprint of ACE+TAO. This porting effort will require removing the use of multiple inheritance in ACE, as well as other minor changes to conform to the embedded C++ features.

Step 3 – Evaluate inter-ORB protocol requirements: The standard CORBA general inter-ORB protocol (GIOP) uses the Internet inter-ORB protocol (IIOP) as its transport protocol [14]. Since IIOP is implemented over TCP certain functionality like adaptive retransmissions, deferred transmissions, and delayed acknowledgments can cause excessive overhead and latency for some signaling applications with real-time QoS requirements. Therefore, we will evaluate alternate Environment-Specific Inter-ORB Protocols (ESIOP) that are customized for the specific environment in which the project is performed (based on the network element selected in step 3.1).

Step 4 – Implement inter-ORB messaging protocols: We will leverage TAO's *pluggable protocols framework* to define an optimized Inter-ORB Protocol that uses is optimized for the particular environment. Three potential modification to the Inter-ORB protocol will be considered during this phase of the project. The first possibility is to use a modified version of the IIOP. The modifications will be minimal and targeted to supporting the ORB within the network element. For example, a lighter weight version of GIOP [19] may be employed to reduce overall message size. This IOP will operate over TCP/IP.

The second set of modification will be centered on developing a new ORB transport protocol that will operate under either GIOP or the lightweight version of GIOP identified above. In this case we will not require TCP/IP to be used as the underlying transport protocol. Instead, we will develop a lightweight ORB transport protocol using UDP or possible AAL5. This will be augmented to provide a byte-stream interface to the ORB messaging component (GIOP).

The third set of modification will consider the creation of a new ORB messaging protocol. This new ESIOP will use existing management and control protocols, such as SNMP or VSI, as the ORB messaging protocol. This will operate over an unreliable, datagram protocol, such as UDP. Naturally, it will also be possible to use the standard GIOP/IIOP protocols to interoperate with other CORBA ORBs.

We plan to conduct extensive empirical tests using these three Inter-ORB protocols.

Deliverables: The deliverables for this step will be the embedded ORB in the Cisco environment and the appropriate Inter-ORB Protocol(s). In addition, we will write papers and technical reports detailing the porting process and selection of IOP for optimal network element management and control.

Task 4: Prototyping and Benchmarking the Real-Time Embedded ORB for Network Switch Control and Management

Activities: This task will bring together the elements developed in tasks 1, 2, and 3 to develop a prototype implementation of the network element management and control agent (NCMA) framework using the TAO embedded ORB ORB. We will perform extensive empirical testing of the embedded ORB and agent framework implementation in the designated Cisco network element. The specific steps involved in this task include the following:

Step 1 – Implement the NCMA framework within the Cisco network switch and Embedded ORB environment: We will first port our NCMA framework to the network element.

Step 2 – Implement NCMA QoS features: Certain control protocols, such as VSI and GSMP, can be used to support different QoS classes required for multimedia support and telecommunication call-setup. QoS-related features we will explore include:

1. Real-time characterization and measurement;
2. Admission control policies at QoS-enabled network elements;
3. Mapping admission controlled method invocations to real-time scheduling primitives, *e.g.*, thread-per-request, thread-per-object, and thread-per-client;
4. Embedded ORB modifications on the network element to implement chosen policies.

A final dimension to be explored is providing a mechanism for associating QoS guarantees to the network element management and control messages, *e.g.*, SNMP or VSI messages. For example, VSI switch configuration messages, such as `connect_commit`, could be assigned a high priority both within the network and the embedded ORB. While routine management messages such as statics collection could have a relatively low, best effort priority. Alternatively, network management event notification traps that signal an error condition could have the highest priority and lowest latency.

As a part of this task we will explore the possibilities and their impact on non-control traffic, connection establishment latencies, error reporting latencies and overall endstation QoS negotiation times.

Step 3 – Conduct fault tolerance experiment with the NMCA framework: In this step, we will conduct experiments using the fault tolerance mechanisms being added to TAO to determine how well they can be used to support NCMA fault tolerance policies.

Deliverables: The deliverables for this task will include a modified implementation of the NMCA framework in the QNX/Neutrino/ENA operating systems and TAO's CORBA IDL compiler, as well as sample applications and benchmarking results. In addition, we will work with the OMG to integrate TAO's pluggable protocols framework and QoS extensions into the CORBA standard.

B Personnel, Schedule, and Budget

The participants in this effort include the following personnel:

1. Faculty member (*i.e.*, Douglas C. Schmidt – Ph.D., Associate Professor, Washington University) at 15% during the period of performance.
2. Senior research associate (*i.e.*, Fred Kuhns, M.S.) at 75% during the period of performance.
3. Graduate student (*i.e.*, Jeff Parsons, M.S. candidate Washington University) at 100% during the period of performance.

The total cost of the proposed 12 month effort is **$98,800**. The following table provides a cost breakdown for this project.

Description	**Amount**
75% full time staff (salary & fringe)	51,000
1 graduate research assistant	35,300
15% Schmidt (salary & fringe)	12,500
total budget	98,800

References

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

[2] L. Cassel, C. Partridge, and J. Westcott. Network Management Architectures and Protocols: Problems and Approaches. *IEEE Journal on Selected Areas in Communications*, SAC-7:1104–1129, September 1989.

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

[4] Zubin D. Dittia, Guru M. Parulkar, and Jerome R. Cox, Jr. The APIC Approach to High Performance Network Interface Design: Protected DMA and Other Techniques. In *Proceedings of INFOCOM '97*, pages 179–187, Kobe, Japan, April 1997. IEEE.

[5] Christopher D. Gill, David L. Levine, and Douglas C. Schmidt. The Design and Performance of a Real-Time CORBA Scheduling Service. *The International Journal of Time-Critical Computing Systems, special issue on Real-Time Middleware*, 2000.

[6] Aniruddha Gokhale and Douglas 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*, 17(9), September 1999.

[7] Michi Henning and Steve Vinoski. *Advanced CORBA Programming With C++*. Addison-Wesley Longman, 1999.

[8] Fred Kuhns, Carlos O'Ryan, Douglas C. Schmidt, and Jeff Parsons. The Design and Performance of a Pluggable Protocols Framework for Object Request Broker Middleware. In *Proceedings of the IFIP* 6^{th} *International Workshop on Protocols For High-Speed Networks (PfHSN '99)*, Salem, MA, August 1999. IFIP.

[9] Fred Kuhns, Douglas C. Schmidt, and David L. Levine. The Design and Performance of a Real-time I/O Subsystem. In *Proceedings of the* 5^{th} *IEEE Real-Time Technology and Applications Symposium*, pages 154–163, Vancouver, British Columbia, Canada, June 1999. IEEE.

[10] Priya Narasimhan, Louise E. Moser, and P. M. Melliar-Smith. Using Interceptors to Enhance CORBA. *IEEE Computer*, 32(7):64–68, July 1999.

[11] Object Management Group. *Fault Tolerance CORBA Using Entity Redundancy RFP*, OMG Document orbos/98-04-01 edition, April 1998.

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

[13] Object Management Group. *Realtime CORBA Joint Revised Submission*, OMG Document orbos/99-02-12 edition, March 1999.

[14] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 2.3 edition, June 1999.

[15] Carlos O'Ryan and Douglas C. Schmidt. Applying a Real-time CORBA Event Service to Large-scale Distributed Interactive Simulation. In 5^{th} *International Workshop on Object-oriented Real-Time Dependable Systems*, Monterey, CA, Nov 1999. IEEE.

[16] 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.

[17] 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.

[18] Guru Parulkar, Douglas C. Schmidt, and Jonathan S. Turner. $\mathrm{a}^{\mathrm{I}}\mathrm{t}^{\mathrm{P}}\mathrm{m}$: a Strategy for Integrating IP with ATM. In *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM)*. ACM, September 1995.

[19] Irfan Pyarali, Carlos O'Ryan, Douglas C. Schmidt, Nanbor Wang, Vishal Kachroo, and Aniruddha 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, May 1999. USENIX.

[20] 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.

[21] Douglas C. Schmidt, David L. Levine, and Sumedh Mungee. The Design and Performance of Real-Time Object Request Brokers. *Computer Communications*, 21(4):294–324, April 1998.

[22] Douglas C. Schmidt, Sumedh Mungee, Sergio Flores-Gaitan, and Aniruddha Gokhale. Software Architectures for Reducing Priority Inversion and Non-determinism in Real-time Object Request Brokers. *Journal of Real-time Systems*, To appear 2000.

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