A CORBA-based Quality-of-Service Management Framework for Distributed Multimedia Services and Applications

James Won-Ki Hong
Dept. of Computer Science and Engineering
POSTECH
jwkhong@postech.ac.kr

Jong-Seo Kim
Network Management Development Team
R&D Center, DACOM
jskim@halla.dacom.co.kr

Jae-Kyu Park
R&D Center
POSDATA
jkpark@mail.posdata.co.kr

Abstract

For distributed multimedia services, it is essential that Quality-of-Service (QoS) is guaranteed system-wide, including end-systems, communication systems and networks. Although many researchers have addressed issues for QoS management, little attention has so far been paid to the QoS management services in distributed multimedia services and applications. To address this deficiency, we have designed a layered model for end-to-end QoS management called the QoS management framework. Our framework, which is CORBA-based, includes a generic QoS MIB for the QoS parameterization of various multimedia services and the services needed to perform various QoS functions. A key component of this framework is the QoS Management Service Object (QMSO) which orchestrates resources at end-points, coordinating resource management across layer boundaries. Services such as translation, monitoring, admission and negotiation are provided by the QMSO. For validating this concept, we have developed a QoS management system for managing and controlling the QoS of a distributed multimedia system called MAESTRO. The QoS management system has been implemented through CORBA objects and provides an interface to multimedia applications, which can be used for dynamic negotiation and renegotiation of QoS by users. Some performance results involved in QoS negotiation and renegotiation are also presented.

Keywords: Quality of Service, QoS MIB, QoS Management, CORBA, Multimedia Services

1. Introduction

The wide spread use of distributed multimedia applications is setting forth a new set of challenges in networking, including management of network resources in order to guarantee Quality-of-Service (QoS) [1, 2, 3]. As users become more familiar with multimedia services, QoS must be approached from the user's perspective as well as from the system or network perspective that has been mainly addressed so far [4, 5, 6, 7]. The user must be given the opportunity to express his requirements for receiving service in terms of QoS parameters familiar to him. In turn, these parameters can then be translated into parameters provided by the underlying systems and networks, satisfying the user's needs [8].

Distributed multimedia services and applications are time-critical and need management support for ensuring the QoS agreed upon by the user and provider [9, 10]. An important aspect of these services is that they require QoS guarantees for the transfer and processing of continuous media data (such as video and audio). Emerging networks such as ATM [11] and the proposed integrated

services with reservations [12, 13, 14] can, up to certain extent, provide guarantees on bandwidth and delay for data transfer. As well, modern computer systems now have sufficient processing power and I/O bandwidth to handle continuous media. In such distributed multimedia computing environments, management has to promote QoS guarantees for each level of the system because the overall QoS depends on the combined QoS of the underlying systems and networks [15]. The end-to-end management, therefore, has to include management capabilities for each layer participating in the delivery of the service, i.e., from the service layer down to the network layer.

Researchers have recently proposed new communication architectures that are broader in scope and cover both network and end-system domains, these architectures differ in several ways, which may be the result of the different communities from which they were developed. The most important difference is related to QoS specifications and parameters, which are fundamental to capturing user-level QoS requirements. Currently, there does not exist any international standard or dominant specification for QoS parameters from the service layer to the network layer. Therefore, designing and providing a set of standard QoS parameters as well as QoS management services, which cover the layers from the service layer to network layer, is the motivation for our work.

Common Object Request Broker Architecture (CORBA) [16] is an emerging distributed object-oriented technology developed by the Object Management Group (OMG). In this paper, we propose a CORBA-based QoS management framework for managing the QoS of distributed multimedia services and applications. We have developed a set of QoS management services, which are needed to monitor and control QoS parameters in distributed multimedia applications. These services are provided by the QoS Management Service Object (QMSO) and have been defined using CORBA IDL. The QMSO orchestrates resources at the end-points, coordinating resource management across layer boundaries. As an intermediary, it hides implementation details from applications and per-layer resource managers. Services such as translation, admission and negotiation are produced by the QMSO to properly configure the system and network to application needs. Configuration is achieved via QoS negotiation resulting in one or more connections through the QMSO. At the heart of the QMSO lies a set of QoS management services that have been modeled using the object-oriented approach and developed as CORBA objects. We provide a set of layered QoS parameters and have defined a generic QoS Management Information Base (MIB) [29] for the QoS parameterization of various multimedia services. The generic QoS MIB can be easily extended to develop MIBs for specific multimedia services.

Earlier, we have developed a CORBA-based distributed multimedia system called MAESTRO [17, 18] that supports the development and operation of distributed multimedia applications. For validation of our QoS management framework, we have developed a QoS management system for managing and controlling the QoS of the MAESTRO distributed multimedia system. The QoS management system has been implemented as a set of CORBA objects and it provides interfaces to multimedia applications, which can be used for dynamic negotiation and renegotiation of QoS.

The organization of the paper is as follows. Section 2 describes QoS concepts, QoS related work, and a comparison of QoS parameters among several previously proposed QoS architectures. Section 3 presents the design of the QoS MIB. Section 4 describes the QoS Management Service Object (QMSO) and the QoS control mechanisms used in the QMSO. Section 5 describes the prototype implementation of our QoS management system for MAESTRO. Section 6 presents some performance results and our experience with the implementation. We summarize our work and discuss some possible future work in Section 7.

2. QoS Concepts and Related Work

QoS is defined as "a set of qualities related to the collective behavior of one or more objects" in [19]. The four main factors determining QoS can be categorized as *negotiation*, *mapping*, *resource reservation*, and *delivery*. QoS negotiation involves resolving the differences between the parties involved over what has been requested and what can be provided. QoS mapping has to be done in various separate steps. The main target remains for the end-system and network to provide the QoS requested by the application. Therefore, application QoS (A-QoS) has to be mapped onto system QoS (S-QoS) or network QoS (N-QoS) and onto protocol-relevant parameters. Resource reservation applies to mainly two areas: the local end-system (normally the operating system) and intermediate system (normally the network). QoS delivery involves carrying out the actions in order to provide the agreed quality of service.

Work on QoS has been done using various targets and scenarios and within a number of projects. In the following subsections, we review a number of distinct approaches that have recently emerged in the QoS research community.

2.1 Int-serv Architecture

The work by the Integrated Services (Int-serv) Group [20] of the Internet Engineering Task Force (IETF) is a significant contribution to providing controlled QoS for multimedia applications over an internetwork. The group has defined a comprehensive architecture and QoS framework used to specify the functionality of internetwork system elements. The behavior of elements, such as routers, subnetworks and end-point operating systems, is captured as a set of services, of which some or all are offered by each element. Each element is QoS-aware and supports interfaces required by the service definition. The concatenation of service elements along an end-to-end data path provides an overall statement of end-to-end QoS. Int-serv currently offers the following optional services: guaranteed service, controlled load service, and best effort service [12, 13, 14].

Flows in an Int-serv architecture are characterized by two specifications: a traffic specification, which is a specification of the traffic pattern that a flow expects to exhibit; and a service request specification, which is a specification of the QoS a flow desires from service elements. The Int-serv architecture, which is restricted to the network but applicable in the end-system too, is comprised of the following components.

- *packet scheduler*, which forwards packet streams using a set of queues and timers.
- *classifier*, which maps each incoming packet into a set of QoS classes.
- *admission controller*, which implements the admission control algorithm to determine whether a new flow can be admitted or denied.
- *reservation protocol* (e.g., RSVP [21]), which is necessary to create and maintain a flow-specific state in the routers along the path of the flow.

2.2 ISO QoS Framework

One early contribution to the field of QoS-driven architectures is the ISO QoS Framework [19], which concentrates primarily on QoS support for Open Systems Interconnection (OSI) communications. The ISO framework broadly defines terminology and concepts for QoS and provides a model that identifies objects of interest to QoS in open system standards. The QoS associated with objects and their interactions is described through the definition of a set of QoS characteristics. Key ISO QoS framework concepts include the following.

- *QoS requirements*, which are realized through QoS management and maintenance entities.
- *QoS characteristics*, which represent the fundamental measures of QoS that have to be managed.
- *QoS categories*, which represent policies governing a group of QoS requirements specific to a particular environment.
- *QoS management functions*, which can be combined in various ways and applied to various QoS characteristics in order to meet QoS requirements.

The ISO QoS framework is made up of two types of management entities (layer specific and system-wide) that attempt to meet QoS requirements by monitoring, maintaining and controlling end-to-end QoS. The task of the policy control function is to determine the policy that applies in a specific layer of an open system. The policy control function models any priority actions that must be performed to control the operation of a layer. The definition of a particular policy is layer-specific and therefore cannot be generalized. Policy may, however, include aspects of security, time-critical communications and resource control. The role of the QoS control function is to determine, select and configure the appropriate protocol entities to meet layer-specific QoS goals. The system QoS control function combines two system-wide capabilities: one to tune the performance of protocol entities and the other to modify the capability of remote systems via OSI systems management. The OSI systems management interface is supported by the systems manager, which provides a standard interface to monitor, control and manage end-systems. The system policy control function interacts with each layer-specific policy control function to provide an overall selection of QoS functions and facilities.

2.3 QoS-A

The Quality of Service Architecture (QoS-A) [1] is a layered architecture of services and mechanisms for the management of quality of service and the control of continuous media flows in multi-service networks. The architecture incorporates the following key notions: flows, which characterize the production, transmission and eventual consumption of single media streams (both unicast and multicast) with associated QoS; service contracts, which are binding agreements of QoS levels between users and providers; and flow management, which provides for the monitoring and maintenance of the contracted QoS levels. The realization of the flow concept demands active QoS management and tight integration of device management, end-system thread scheduling, communications protocols and networks.

In functional terms, the QoS-A is composed of a number of layers and planes. The upper layer consists of a distributed applications platform augmented with services to provide multimedia communications and QoS specification in an object-based environment. Below the platform level is an orchestration layer that provides jitter correction and multimedia synchronization services across multiple related application flows. Supporting this is a transport layer that contains a range of QoS configurable services and mechanisms. Below this, an internetworking layer and other lower layers form the basis for end-to-end QoS support. QoS management is realized in three vertical planes in the QoS-A.

2.4 QuAL

An important contribution by Florissi [22] is the development of a Quality Assurance Language (QuAL) for the specification of QoS constraints on underlying computing and communication platforms. QuAL eases the management of QoS for distributed multimedia computing and

communication applications. QuAL provides language-level abstractions for the specification, negotiation, and management of communication and processing QoS constraints. The communication QoS constraints include network level constraints (such as maximum propagation delay) and application level ones (such as message delivery rate and synchronization of streams). The processing QoS constraints specify the execution timing demands of activities (such as deadlines to process messages). QuAL exposes application developers to a single, generic abstraction for QoS specification that is independent of the underlying service providers (transport layer and operating system). QoS parameters specified are used by the language runtime to allocate the communication and processing resources that can best deliver the QoS required.

2.5 QoSMA

The QoS management architecture (QoSMA) [23] follows the vertical approach and uses QoS objects (that must be particularized for each layer of the architecture). This architecture is made up of four layers. There are three external entities – user, service provider, and network provider - representing the parties that most often play the basic roles in the deployment of multimedia services. Each entity may interact with the corresponding layer and interactions may or may not exist, depending on the context. Users are more likely to interact only with the service layer, however, interaction with the underlying layers may be allowed, for example, if the user is the programmer of the application and has access (even partially) to the system resources. The network provider is the one who interacts with the lower layers of the architecture, i.e., the layers that deal with the communication part of multimedia services.

2.6 QuO

Quality of Service for CORBA Objects (QuO) [24] develops a cohesive framework for constructing adaptable applications by introducing concepts for quality of service of object access and by providing mechanisms which can be used to integrate these concepts into emerging applications. The initial focus of this work has been on adapting to and managing network communication resources. However, the approach taken also lends itself to a broader definition of quality of service that incorporates other system properties such as fault tolerance, security, and end–to-end performance. The QuO concept integrates the currently dispersed knowledge about system properties into a cohesive framework so that both the base value and variance of these system properties can be improved. This is accomplished by using traditional operating system techniques and concepts; the runtime software is used to provide an easier to use and more manageable abstract resource than is provided by the existing infrastructure.

QuO has been developed to support QoS in the CORBA layer. QuO extends the CORBA functional Interface Description Language (IDL) with a QoS Description Language (QDL). QDL specifies an application's expected usage patterns and QoS requirements for a connection to an object. The QoS and usage specifications are at the object level and not at the communication level. An application can have many connections to the same object, each with different system properties. QDL allows the object designer to specify QoS regions, which represent the status of the QoS agreement for an object connection. The application can adapt to changing conditions by changing its behavior based on the QoS regions of its object connections. Finally, QuO provides mechanisms for measuring and enforcing QoS agreements and for dispatching handlers when the agreements are violated. These mechanisms can help distributed applications be more predictable and adaptive even when end-to-end guarantees cannot be provided.

2.7 The Heidelberg QoS Model

The HeiProject [25] at IBM's European Networking Center in Heidelberg has developed a comprehensive QoS model that provides guarantees in the end-systems and network. The communications architecture includes a continuous media transport system (HeiTS/TP) that provides QoS mapping and media scaling. Underlying the transport layer is an internetworking layer based on ST-II, which supports both guaranteed and statistical levels of service. In addition, the network supports QoS-based routing (via a QoS finder algorithm) and QoS filtering. The key to providing end-to-end guarantees is HeiRAT (resource administration technique). HeiRAT comprises a QoS management scheme that includes QoS negotiation, QoS calculation, admission control and QoS enforcement, and resource scheduling. The HeiRAT scheduling policy used in the supporting operating system is a rate-monotonic scheme whereby the priority of a thread performing protocol processing is proportional to the message rate accepted.

3. QoS MIB

In this section, we present a generic Quality-of-Service (QoS) management information base (MIB) for multimedia services. We call it a generic QoS MIB because it contains QoS related management information common to most (if not all) multimedia services. Thus, the QoS MIB can be used to monitor and control QoS values in a multitude of multimedia services. However, if specific QoS management information must be obtained in order to manage a particular multimedia service, adding the service-specific QoS management information will easily accomplish this. In designing our QoS MIB we have utilized much of the previous work done by several IETF working groups. The recent effort carried out by the IETF to define MIBs for integrated services can be considered similar to our effort. It includes “intSrv MIB” [26] and “intSrvGuaranteed MIB” [27]. In addition, a number of RFCs (from RFC 2210 to RFC 2216, inclusively) are defined for integrated services.

Traditional QoS (as in the ISO standards) mostly referred to measures at the network layer of a communication system and QoS enhancement was achieved by introducing QoS into transport services. For multimedia services, however, the notion of QoS must be further extended, as many other services contribute to end-to-end service quality. To discuss QoS then, we need a layered model of the multimedia service with respect to QoS [28]. We assume throughout this section the model shown in Figure 1.

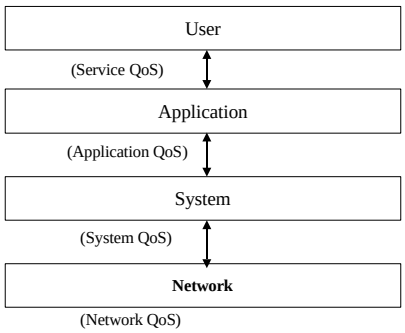

Figure 1. Layered QoS Model

The internal architecture of a multimedia service consists of three layers: application, system (including operating system services), and network. Above the application on the client side resides

a human user. This layered structure implies the introduction of service QoS, application QoS, system QoS, and network QoS. The QoS MIB includes mainly objects, which represent values of each layered QoS parameter. These managed objects are organized together into four logical groups (as illustrated in Figure 2): the service group, the application group, the system group, and the network group [29]. In each group, we define objects maintaining QoS related information.

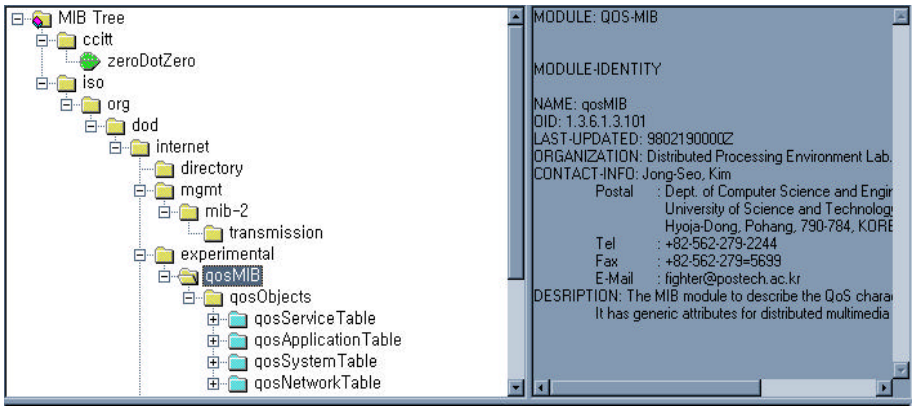

Figure 2. QoS MIB Tree

The service layer QoS parameters specify the service quality which the user wishes to see or hear (e.g., TV quality of video, telephone quality of audio). The service quality, however, is hard to quantify, its evaluation is subjective and user dependent. The service QoS parameters can be characterized according to the temporal quality, where the user perceives (listens/views) the playback rate of the audio/visual media, and spatial perceptual quality, where the user perceives the spatial details of the individual audio sample or image frame.

The application layer QoS parameters describe requirements for application services possibly specified in terms of media quality, which includes the media characteristics, transmission characteristics, and media relations. The parameterization also includes an application-oriented specification of the required transmission characteristics for end-to-end delivery (e.g., end-to-end delay bounds). The media relations specify the relationships among the media streams. Synchronization skew represents an upper bound on time offset between two streams in a single direction. This information can be used for a finer granularity scheduling decision of multimedia streams than the sample rate information of periodic streams provides. The application layer QoS parameters can be mapped from the service layer or directly characterized by users. They can be parameterized differently according to media or service type.

System QoS parameters describe requirements of the communication services and operating system (OS) services defined by the application QoS. The OS services require the following resources: processing time, secondary storage, and memory buffer. The OS resources are needed by the application-layer and network-layer tasks to handle the input/output of media and for the sending/receiving connections. The definition of QoS parameters at the system layer requires mainly CPU scheduling, memory management for buffering, and secondary storage space.

Network layer QoS parameters may be specified in three domains of interest. The first domain includes throughput specification such as packet size, packet rate, and burstiness. The second domain includes flow specification such as inter-arrival delay, round-trip delay and packet loss rate. The last domain includes performance specification such as packet ordering and priorities. We have

defined the network QoS parameters on the basis of intSrv MIB [26] defined by IETF. In addition, we have used a number of other RFCs and referred to RSVP [21], QoS-A [1] and QoS Broker [30] for our definition. The main difference between our MIB and IETF QoS-related MIBs is that we focus more on the higher-level including the service and application level aspects of QoS for supporting distributed multimedia services.

4. QoS Management Services

To provide applications with end-to-end QoS guarantees, network resource management alone is not sufficient, particularly when end-points become more sophisticated (e.g., workstations that are equipped with a rich set of multimedia devices and support multiprocessing and multiple users). This requires balancing resources among the applications, system, and network at the end-points as well as between the end-points and the network. To provide QoS guarantees, we designed and implemented the QoS Management Service Object (QMSO), which provides local balance (local allocation of resources) at the end-points and which cooperates with the network resource management to achieve global balance (global allocation of resources) as shown in Figure 3.

Local balance includes communication of QoS parameters among application, system, and network components, testing for availability of end-point resources based on QoS requirements, and reservation/allocation of these resources. For global balance, the QMSO negotiates between the end-points. In the negotiation process, the QMSO assumes different roles (sender and receiver) to distinguish between the participating partners.

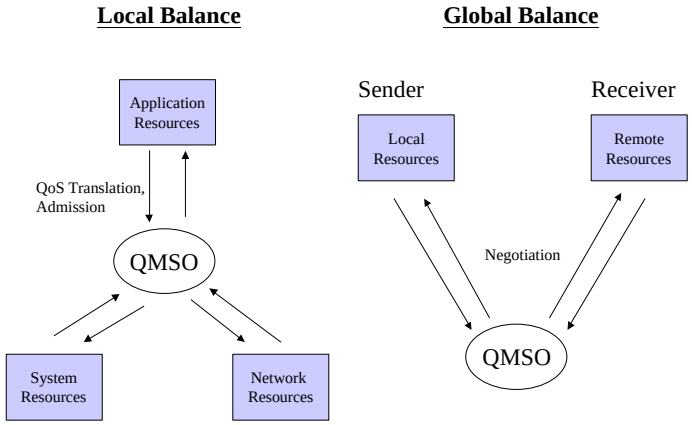

Figure 3. Local and Global Balances Using the QMSO

The The service contract negotiated by the QMSO is stored in shared profiles in the form of a QoS MIB so that application and network tasks can react according to resource availability. In the next subsections, we discuss individual QoS services and describe the architecture of the QMSO.

4.1 QMSO Services

In this subsection, we describe the individual services – QoS specification, QoS monitoring, QoS adaptation, QoS mapping, admission control, and negotiation/renegotiation – provided by the QMSO.

4.1.1 QoS Specification

To guarantee QoS in multimedia services, QoS is represented in the form of certain QoS parameters. The value of the QoS parameters can be acquired from device specifications, off-line testing, or on-line testing. We use all of these methods for QoS specification, therefore, a brief explanation of each of them is valuable at this point.

• Device Specification: A video card and its accompanying software provide a description of the possible frame rates and frame sizes that the card and the driver can support (e.g., XVideo 700 Parallax Video provides 640 x 482 pixel frames at 30 frames/second). The multimedia application may take these parameters as the QoS specification. However, parameters are often not realistic since the end-to-end QoS depends on many different factors such as: (1) the application software running on the client/server sides, (2) the transport protocol stack used by the multimedia service, (3) the CPU utilization by other applications during runtime of the actual application; and (4) the underlying network.

• Off-line Testing: Another approach to QoS specification is to run extensive off-line tests to pre-determine the QoS parameters of the QoS MIB. The QoS parameters are then stored in configuration files and retrieved when the negotiation phase begins. The off-line testing can be used to form a map of the performance of the system at different load levels. In addition, admission control can be used to keep the loads at acceptable levels for the given performance requirements.

• On-line Testing: In the environment of adaptive applications on top of shared OSs and networks, the two QoS specification methods mentioned above do not provide a good estimate of a realistic and possible QoS for the application negotiation protocols. Consequently, we have applied a probe-based algorithm [30, 34], which is an on-line testing method for QoS specification. This algorithm is based on probes performed at the beginning of the call set up phase and includes the following actions: (1) determination of the application QoS of a continuous medium as a statistical guarantee, (2) determination of the degradation point at the client side when system performance starts to severely degrade due to buffer problems and mismatched rates between the server and the client, and (3) provision of QoS suggestions for the negotiation and transmission phase to avoid severe degradation.

4.1.2 QoS Monitoring

QoS monitoring is an important part of our system. Since monitoring can add overhead during multimedia transmission, it is preferable that it is flexible. This flexibility means that most of the monitoring variables should be optional and monitoring should be able to be turned on and off. In general, QoS monitoring consists of two modes: a query mode and a report mode. The former requests a status report about resource utilization and QoS guarantees; the latter regularly reports the QoS and resource status. In our system, we use the report mode. The human user gets the status report and the monitoring service at the client application level regularly reports the status. Monitoring at the client side includes a supervisor function to continuously observe that the processed QoS parameters do not exceed their negotiated values.

4.1.3 QoS Mapping

The mapping between layers is performed in the QoS management plane by the QoS mapper. The

QoS mapper maps the QoS parameters of one layer onto the QoS parameters of other layers and vice versa (bi-directional translation). The mapping includes at least the three activities listed below. Figure 4 illustrates the QoS mapping activities and an example mapping.

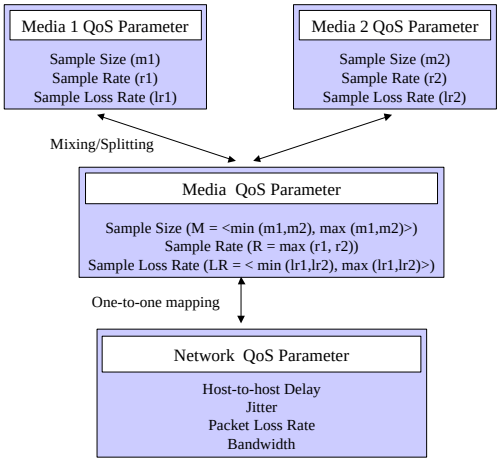

Figure 4. QoS Mapping

• One-to-one mapping involves a translation between the network connection quality and the medium quality.

• Mixing involves multiplexing different media into a single stream that will be sent through a single network connection. This implies that the different media qualities have to be merged into a one new media quality specification. After mixing of the media qualities, a one-to-one translation occurs between the resulting media quality and the network QoS for a connection. It is important to point out that the resulting media quality is the union with precedence of the media qualities being integrated. Therefore, mixing should be done on media having similar QoS requirements, otherwise a stream with unrealistic QoS requirements will result. Also, the mapping is bi-directional, so ambiguities can also occur in this case. Therefore, the QoS mapper passes to the application several possibilities and lets the application/user decide which medium will suffer in quality. In a more sophisticated system, a rule-based QoS mapper can be deployed which will make decisions based on rules given by the user a priori.

• Splitting involves demultiplexing a medium stream into several streams, which will be transmitted through several connections. This occurs when the media stream carries different kinds of information (e.g., in a MPEG compressed video stream). Since the interframe media quality specification includes the intraframe specification, the QoS mapper can perform one-to-one mapping immediately between the intraframe component specification and the network QoS.

It is important to note that the mixing and splitting of streams are very difficult issues and require much further research.

4.1.4 Admission Control

Admission control is an essential element to achieve guaranteed services. For distributed multimedia communications systems, each resource along the path(s) between source(s) and sink(s)

must monitor its availability. The QMSO performs admission control at both the system and network layers. For ease of implementation, we adapt a non-preemptive scheduling algorithm because non-preemptive algorithms are relatively easy to implement. One drawback is that a high priority message can be blocked by a long low priority message [31].

The admission service performs four tests at the system level: a device quality test, a local schedulability test, an end-to-end delay test and a buffer allocation test. These tests check the multimedia devices and the availability of system resources for the multimedia service. The device quality test compares the configuration parameters of the multimedia devices with the specified application QoS requirements. For example, if a video device can provide a maximal frame rate of 15 frames/second and the user specifies the application QoS sample rate of 30 frames/second, then the admission service either rejects the requested QoS and waits for correct user input, or if "falls back" to the feasible QoS and informs the user of the change. The local schedulability test takes the system QoS parameters, which specify the application tasks for processing multimedia streams, and checks if the tasks are schedulable. The behavior of the application tasks allows us to test the tasks as if they would be scheduled using the Earliest Deadline First (EDF) policy [30]. If the result of the schedulability test is satisfactory, the task sequence is assigned according to the deadline of each task (highest priority is assigned to the earliest deadline). If there are input and output tasks with the same deadline, the input tasks receive greater precedence than the output tasks. The end-to-end delay (EED) test consists of two steps. On the client side, the test takes the duration of the local client application tasks and checks them against the specified QoS EED bound. Here, we make sure that the tasks, although schedulable, do not violate the EED requirement. On the server side, the processing times of application tasks, network tasks over connections, and the actual network latency are taken into account. The buffer allocation test determines if there is enough memory space for the ring buffers assigned to multimedia devices in real memory and smooths the traffic jitter. Smoothing the traffic jitter is required when the measured EED is smaller than the requested EED.

The admission service at the network level performs tests on network resources such as a throughput test, rate control test, and network EED test. The throughput test controls the assignment of bandwidth to individual connections. The network host interface and its device driver determine the upper bound of available aggregate throughput at the end-point. For example, in our system the host network interface provides a transmission rate of 100 Mbps (note that this is a theoretical upper bound since an actual 100 Mbps Fast Ethernet interface typically does not yield such bandwidth due to internal bottlenecks). Hence, any throughput requested for the sending or receiving connections is checked against the 100 Mbps bound. The rate control test checks the number of network packets per second, moved from/to user space to/from the network host interface, against a certain bound. The end-to-end delay test checks the duration of network tasks at the end-points against the required end-to-end delay bound. The same approach as in the system layer must be considered here. The buffer allocation test is needed if the network tasks are to queue the incoming/outgoing packets.

Admitted Resources	Admission Tests
Bandwidth	Sum (throughput) <= bound (e.g., 100Mbps)
Rate Control	Sum (the number of network packets per second) <= bound
EED	Sum (all processing time) <= EED delay bound
Buffer	Allocated buffer size <= upper bound of memory

Table 1. Admission Test in Network Level

4.1.5 Negotiation and Renegotiation

The QoS parameters are exchanged between server and client through peer-to-peer negotiation and layer-to-layer negotiation. The peer-to-peer negotiation is separated into two levels: application QoS negotiation and network QoS negotiation. We split the negotiations because, during the application negotiation, we allow not only negotiation of application QoS parameters, but also negotiation of additional information relevant to the applications, such as images, position, etc. This approach allows the application to negotiate specific goals without reserving/allocating/holding shared network resources. The network negotiation of QoS is an exchange of negotiation messages about the traffic quality on different connections.

If the negotiation at the application level succeeds, the QMSO initiates application-to-network negotiation. From the view of the application it is a bi-directional negotiation, which means that application QoS parameters are forwarded (through the QoS Translator) to the network, and the network layer QoS manager negotiates the QoS values within the local system and may consequently change them. An important part of the negotiation process is the translation of QoS parameters done by the QoS Translator. Renegotiation is performed during the transmission phase. Here, the user/application or network can store their request for renegotiation and one possible parameter to change. If a request for change is specified, the QMSO is invoked and it changes the contract. If possible, this is done as a background task. Figure 5 shows the QoS control procedure in the QMSO.

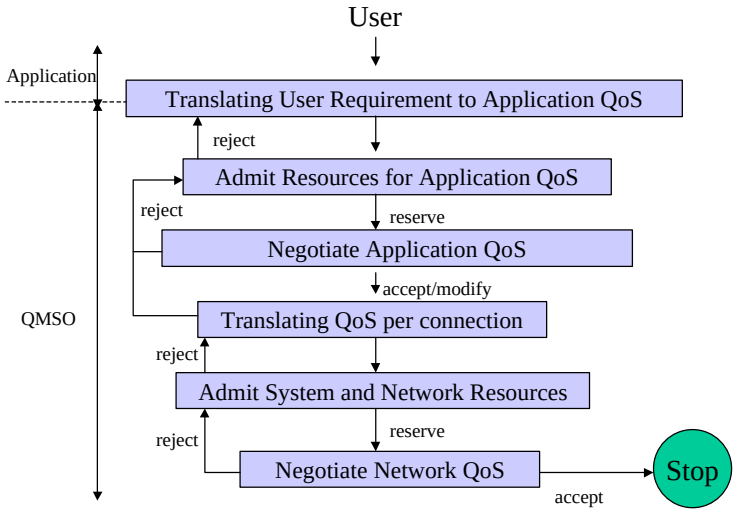

Figure 5. QoS Control Procedure in the QMSO

4.2 QMSO Architecture

The QMSO is classified into four sets of services, which include service-layer QoS management, application-layer QoS management, system-layer QoS management and network-layer QoS management. These services are used by the multimedia application for supporting QoS guarantees. Figure 6 illustrates the services included as part of the QMSO, which is composed of a number of layers and planes.

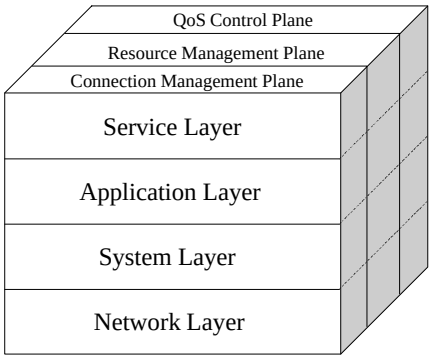

Figure 6. The Structure of QoS Management Services

The service layer offers the user the option to specify quality requirements and priorities for the employed media services. It includes a QoS specification service in order to specify the user's requirements for various media services in a uniform way. It also provides the necessary communication primitives for establishment and disconnection of multimedia sessions and translates the underlying QoS parameters to user perceptible parameters. The application layer is responsible for translating user requirements for multimedia services into a set of QoS, application, system, and network requirements. It executes various control mechanisms to compare and control the user-specified quality with the currently available quality. The controlling mechanism operates on different layers (service, application, system and network layers). The system layer functions in translating application layer QoS parameters into underlying layer QoS parameters and in reverse translation. It also has a QoS control mechanism and is responsible for end-system resource management: CPU scheduling, and memory and I/O management. The network layer is responsible for translating upper layer QoS parameters into network QoS parameters and for QoS control.

The connection management plane manages the establishment of a multimedia session based on a user supplied profile and it is made up of layer-specific connection managers that bind multimedia processing units (MPUs) at each layer in order to meet end-to-end connectivity. The resource management plane is responsible for resource management and monitoring. It performs both admission testing and resource reservation at every level in the end-system. In this instance, it actively measures the CPU usage and periodically informs the CPU scheduler of the utilization. The QoS control plane is the central arbitrator of end-to-end QoS. It is comprised of layer-specific QoS managers that negotiate resources with peer QoS managers and it maintains the internal state associated with application specific QoS. It translates user requests for multimedia flows into a set of QoS parameters. The QoS mapping is based on the definition of QoS MIB, described in the previous section. An important function of the QoS control plane is to monitor layer-specific QoS and report any QoS violations of the contracted profile directly to the multimedia applications.

5. Prototype Implementation

In Section 3, we presented a MIB definition for the QoS management of distributed multimedia services. Using this and the QoS management services defined in Section 4, we have developed a prototype QoS management system to validate our concepts. The target system we wish to manage is MAESTRO [17, 18], which is a CORBA-based distributed multimedia system developed earlier in our lab. MAESTRO is an object-oriented, distributed multimedia system, whose goal is to provide the multimedia services needed to easily develop and operate a variety of multimedia

applications. MAESTRO is composed of a number of essential distributed service objects including a Name Service Object (NSO), Communication Service Object (CSO), Session Service Object (SSO), and Storage and Retrieval Service Object (SRSO).

The Management Service Object (MSO) has been developed to manage and control service objects in MAESTRO [29]. For management purposes, the MSO interacts with managed service objects via the Management Interface Object (MIO). We have extended the MSO and the MIO and developed a QoS Management Service Object (QMSO) and a QoS Management Interface Object (QMIO). The QMIO is a specialized version of the MIO used for accessing QoS information related to multimedia communication from the SSO and CSO.

In the QoS management system, the QMSO performs QoS functions (including QoS control, QoS mapping, and resource monitoring and control). When the user starts a multimedia application (such as video conferencing), the user is asked to provide the QoS requirements. In our prototype, this is accomplished by using the user-level QoS manager tool, which can be used to specify the quality of video, video encoding, window size, etc. (as shown in Figure 7). The QMSO then translates these user requirements into application-specific parameters and into QoS requirements for the underlying resources. In addition, the QMSO monitors and controls the available host and network resources. Our QoS management service has been implemented on a CORBA platform, IONA Orbix 2.2 [32], because MAESTRO was developed on the same platform.

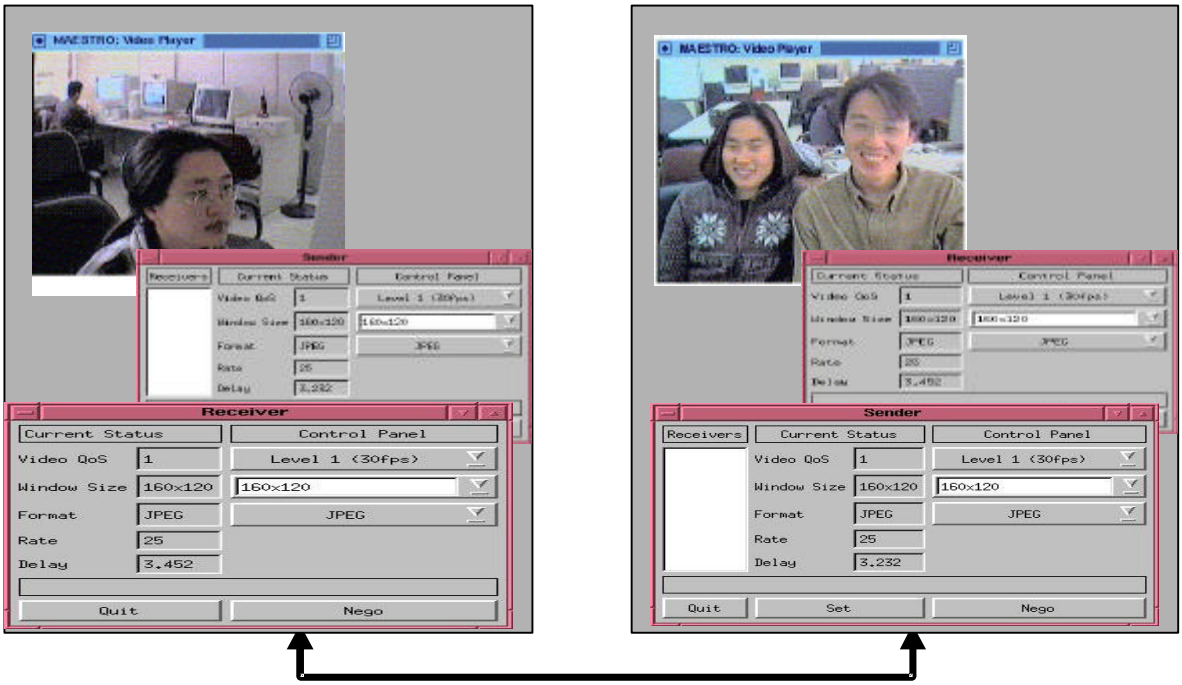

Figure 7. Negotiation in Video/Audio Conferencing using QoS Manager Tool

Figure 7 shows the video/audio conferencing tool with the developed QoS guarantees, using and operating on MAESTRO. The QoS management tool has been implemented using C++ and an X Window/Motif library for user interface, and a Sun Video/Audio Card for capturing video and audio. With this tool, users can control the transmission and reception of video and audio data separately. Further, users can specify media service quality and view the sender's supported service quality. If the sender's service quality is equal to that which the users wish to receive, service is initiated, but if

not, the negotiation process takes place. For advanced users, we also provide an advanced QoS manager tool, which allows more sophisticated specification of QoS requirements.

The users participating in a video conferencing session can dynamically change the QoS parameter settings during a session. Initially, the receiver sets the QoS parameters using the manager tool and then sends a negotiation request to the sender. The sender may then accept the requested parameters and acknowledge the receiver's request. If the sender cannot accept the request, the sender sends a rejection message to the receiver indicating the best QoS the sender can offer. Based on this, the receiver can repeat the negotiation process with more acceptable QoS parameters. During a multimedia session, the receiver may attempt to renegotiate with the sender for a different (higher or lower) quality of service. The renegotiation process is the same as that of negotiation.

6. Performance

In this section, we discuss some issues involved in providing a QoS mechanism that attempts to enhance the quality of service for multimedia system users. We also present some performance results of the prototype implementation of our QoS management system.

In our implementation, the QoS management service provides a five-level scale to define the quality of a multimedia service, allowing the user the opportunity to specify one of these levels as a way to express the requirements. In general, the higher the quality level a user chooses, the greater the user's system requirements must be. For example, when the user selects to receive video quality at 30 frames per second, this requires more bandwidth, CPU resource and buffer space than selecting video quality of 15 frames per second. What the QoS management service can do (compared to without the service) is to provide higher quality whenever possible given the resources available.

Obviously, our QoS management system does incur some overhead. One major overhead involved is the instrumentation of functions that can provide QoS parameters to existing systems. We have minimized this overhead by carefully instrumenting the QMIO into existing CORBA objects using the inheritance feature of the object-oriented paradigm. Another overhead involved is the extra resources and timing involved in exchanging messages during negotiation and renegotiation periods. However, these messages consist of small amounts of non-multimedia negotiation protocol data and thus, this does not incur any serious overhead.

Table 2 shows the extra times taken for negotiation under various conditions. The experiments were carried out between two workstations connected on a 10 Mbps Ethernet network. The first workstation (W1) is a Sun Ultra Sparc 2 running Solaris 2.5 and the second workstation (W2) is a Sun Ultra Sparc 1 running Solaris 2.5. "W1 (light)" means that the CPU of W1 was lightly loaded and "W1 (heavy)" means that the CPU of W1 was heavily loaded (greater than 1.0) when the measurements were taken. We also performed the experiments both under light network load (less than 10% network utilization) and under heavy network load (greater than 30% network utilization). "W1 to W1" means that both the sender and the receiver were in a single workstation (in this case the network load does not matter since the communication does not take place over the network). The performance results show that the CPU load on the workstations put a greater effect on the performance of the negotiation than the network load. It is also important to note that the performance overhead for negotiation is fairly minimal.

	Light Network Load	Heavy Network Load
W1 (light) to W2 (light)	1.32 msec	2.68 msec
W1 (light) to W1 (light)	1.12 msec	1.12 msec
W1 (heavy) to W1 (heavy)	1.57 msec	1.57 msec
W1 (light) to W2 (heavy)	1.49 msec	3.64 msec
W1 (heavy) to W2 (heavy)	5.95 msec	8.87 msec

Table 2. Performance Results of QoS Negotiation

We believe the gain in providing QoS management service outweighs the overhead involved, however, there is still a lot of room for improvement. This is especially true when resources (both system and network) are scarce but one wishes to provide a higher quality of service to users.

7. Conclusions and Future Work

In this paper, we have proposed a CORBA-based QoS management framework for managing the QoS of distributed multimedia services. A generic QoS MIB has been defined for the QoS management of various multimedia services. The definition and adoption of the QoS MIB will allow for uniform and standardized QoS management for information in multimedia services and it can be easily extended to develop MIBs for specific multimedia services. The QoS MIB can be easily used to integrate QoS management into standard network management frameworks (such as SNMP and CMIP [33]). The QoS MIB provides a set of layered QoS parameters and can be used as standard QoS parameters for managing QoS in multimedia services. We have also described the QoS management services for distributed multimedia services. They are classified into four sets of services, which include service-layer QoS management, application-layer QoS management, system-layer QoS management and network-layer QoS management. These services are used by the multimedia application for supporting QoS guarantees. These services are included as part of the QoS Management Service Object (QMSO), which is composed of a number of layers and planes.

We have by no means completely worked out the details of bi-directional QoS translation, admission tests or negotiation among system elements. A better understanding of user requirements is necessary. As well, system layer QoS parameters need to be refined. Additional parameters for each layer should be taken into account and more work is needed in defining accurate QoS control mechanisms. Also, QoS mapping, particularly mixing and splitting of streams, is very difficult and requires much further research. For example, when mixing different quality multimedia streams into a single stream, it is difficult to preserve the original qualities and thus almost impossible to recover them when demultiplexing.

Another area of future work is to compare the overhead of using an object-oriented system such as CORBA to do the management versus a non-object-oriented system such as RTP/RTCP [35] based control.

The major contribution we have made in our work thus far is to provide a QoS capability for a distributed multimedia service management framework, particularly, for a CORBA-based one. We have demonstrated that this kind of approach does work even in an environment where QoS is not guaranteed at the network level.

References

[1] A. Campbell, G. Coulson, F. Garc, D. Hutchison, and H. Leopold, "Integrated Quality of Service for Multimedia Communications," *Proc. IEEE INFOCOM'93*, San Francisco, April 1993, pp. 732-739.

[2] A. A. Lazar and G. Pacifici, "Control of resources in broadband networks with Quality of Service guarantees," *IEEE Communications Magazine*, vol. 29, no. 10, October 1991, pp. 66-73.

[3] M. Woo et al, "A synchronization framework for communication of pre-orchestrated multimedia information," *IEEE Network*, vol. 8, no. 1, January 1994, pp. 52-61.

[4] N. Yamanaka et al, "Full-Net: A flexible multi-QoS ATM network based on a logically con- figured VC-network," *Proc. of ISS'95*, Berlin, April 1995, vol. 2, pp. 195-199.

[5] S. Cronjaeger and P. Vindeby, "Translation of multimedia user requirements into ATM switching requirements by using appropriate transport protocols," *Proc. of ISS'95*, Berlin, April 1995, vol. 1, pp. 180-184.

[6] B. Gadher et al, "A distributed broadband metropolitan network for residential multimedia applications," *Proc. of ISS'95*, Berlin, April 1995, Vol. 2, pp. 190-194.

[7] J. Hall et al, "Customer requirements on teleservice management," *Integrated Network Management IV*, Ed. A.S. Sethi et al., Chapman & Hall 1995, pp. 143-155.

[8] J. Jung and D. Seret, "Translation of QoS parameters into ATM performance parameters in B-ISDN," *Proc. of IEEE INFOCOM'93*, San Francisco, March 1993, pp. 748-755.

[9] F. Georges, "Performance evaluation & management of interconnected networks," *Proc. of SBT/IEEE International Telecommunications Symposium*, Rio de Janeiro, August 1994, pp. 322-326.

[10] F. Somers and M. Edholm, "Intelligent resource dimensioning in ATM networks," *Proc. of ISS'95*, Berlin, April 1995, vol. 2, pp. 62-66.

[11] S. E. Minzer, "Broadband ISDN and asynchronous transfer mode (ATM)," *IEEE Communications Magazine*, vol. 27, no. 9, September 1989, pp. 17-24.

[12] J. Wroclawski, "The Use of RSVP with Integrated Services," RFC 2210, September 1997.

[13] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211, Sept. 1997.

[14] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service," RFC 2212, Sept. 1997.

[15] D. Hutchison et al, "Quality of service management in distributed systems," *Network and Distributed Systems Management*, Ed. M. Sloman, Addison-Wesley 1994, pp. 273-302.

[16] OMG, "The Common Object Request Broker: Architecture and Specification Revision 2.0," July 1995.

[17] T. H. Yun, J. Y. Kong and J. W. Hong, "A CORBA-based Distributed Multimedia System," *Proc. of the Fourth Workshop on Distributed Multimedia Systems*, Vancouver, Canada, July 1997, pp. 1-8.

[18] J. W. Hong, T. H. Yun, J. Y. Kong, and Y. M. Shin, "A Flexible and Reliable Distributed Multimedia System for Multimedia Information Superhighways," *Malaysian Journal of Computer Science*, Vol. 10, No. 2, December 1997, pp. 1-16.

[19] ISO, "Quality of Service Framework," ISO/IEC JTC1/SC21/WG1 N9680, International Standards Organization, UK, 1995.

[20] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, 1994.

[21] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," RFC 2205, Sept. 1997.

[22] P. Florissi, "QuAL: Quality Assurance Language," *Technical Report* CUCS-007-94, Columbia University, March 1994.

[23] M. Alfano, "A Quality of Service Management Architecture (QoSMA): A preliminary study," *Technical Report* TR-95-070, International Computer Science Institute, December 1995.

[24] J. Zinky, D. Bakken, and R. Schantz, "Architectural Support for Quality of Service for CORBA Objects," *Theory and Practice of Object Systems*, vol. 3, no. 1, April 1997, pp. 55-73.

[25] D. Hehmann, R. Herrtwich, W. Schulz, T. Schuett and R. Steinmetz, "Implementing HeiTS: Architecture and Implementation Startegy of the Heidelberg High Speed Transport System," *Proc. of Second International Workshop on Network and Operating System Support for Digital Audio and Video*, IBM ENC, Heidelberg, Germany, 1991.

[26] F. Baker, J. Krawczyk and A. Sastry, "Integrated Services Management Information Base using SMIv2," RFC 2213, Sept. 1997.
[27] F. Baker, J. Krawczyk and A. Sastry, "Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2," RFC 2214, Sept. 1997.
[28] K. Nahrstedt and R. Steinmetz, "Resource management in networked multimedia systems," *IEEE Computer*, May 1995, pp. 52-63.
[29] J. Y. Kong, J. W. Hong, J. T. Park and D. J. Kim, "A CORBA-Based Management Framework for Distributed Multimedia Services and Applications," *Proc. of the Distributed Systems: Operations and Management*, Sydney, Australia, October 1997, pp. 132-144.
[30] K. Nahrstedt and J. Smith, "The QoS Broker," University of Pennsylvania, Philadelphia, 1995.
[31] R Carmo et al, "Real-Time Communication Services in a DQDB Network," *Proc. of Real-Time Systems Symposium*, San Juan, Puerto Rico, December 1994, pp. 249-258.
[32] IONA, *Orbix 2*, IONA Technologies, Release 2.2, March 1997.
[33] U. Black, *Network Management Standards: SNMP, CMIP, TMN, MIBS, and Object Libraries*, McGraw Hill, November 1994.
[34] K. Nahrstedt, A. Hossain and S. Kang, "A probe-based algorithm for qos specification and adaptation," *Proc. of 4th IFIP Workshop on Quality of Service*, Paris, France, March 1996, pp. 89-100.
[35] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Audio-Video Transport Working Group, RFC 1889, January 1996.

List of Acronyms

A-QoS	Application QoS
ATM	Asynchronous Transfer Mode
CMIP	Common Management Information Architecture
CORBA	Common Object Request Broker Architecture
CSO	Communicaiton Service Object
EDF	Earliest Deadline First
EED	End-to-End Delay
IDL	Interface Description Language
ISO	International Organization for Standardization
ITU-T	Interantional Telecommunicaitons Union – Telecommunications Standardization Sector
Int-serv	Integrated Services
Int-serv Group	The Integrated Services Group
MIB	Management Information Base
MIO	Management Interface Object
MPU	Multimedia Processing Units
MSO	Management Service Object
N-QoS	Network QoS
NSO	Name Service Object
OMG	Object Management Group
OSI	Open System Interface
QDL	QoS Description Language
QMIO	QoS Management Interface Object
QMSO	QoS Management Service Object
QoS	Quality of Service
QoS-A	QoS Architecture
QoSMA	QoS Management Architeture
QuAL	Quality Assurance Language
QuO	Quality of Service for CORBA Objects
S-QoS	System QoS
SNMP	Simple Network Management Protocol
SRSO	Storage Retrieval Service Object
SSO	Session Service Object