

January 1995

# End-Point Resource Admission Control for Remote Control Multimedia Applications

Klara Nahrstedt
*University of Pennsylvania*

Jonathan M. Smith
*University of Pennsylvania*, jms@cis.upenn.edu



University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-95-18.

# End-Point Resource Admission Control for Remote Control Multimedia Applications


## Abstract

One goal in certain classes of networked multimedia applications, such as full-feedback remote control, is to provide end-to-end guarantees. To achieve guarantees, all resources along the path(s) between the resource(s) and sink(s) must be controlled. Resource availability is checked by the admission service during the call establishment phase. Current admission services control only network resources such as bandwidth and network delay. To provide end-to-end guarantees, the networked applications also need operation system resources and I/O devices at the endpoints. All such resources must be included in a robust admission process. By integrating the end-point resources, we observed several dependencies which force changes in admission algorithms designed and implemented for control of a single resource. We have designed and implemented the *multi-level admission service* within our *Omega architecture* which controls the availability of end-point resources needed in remote control multimedia applications such as telerobotics.



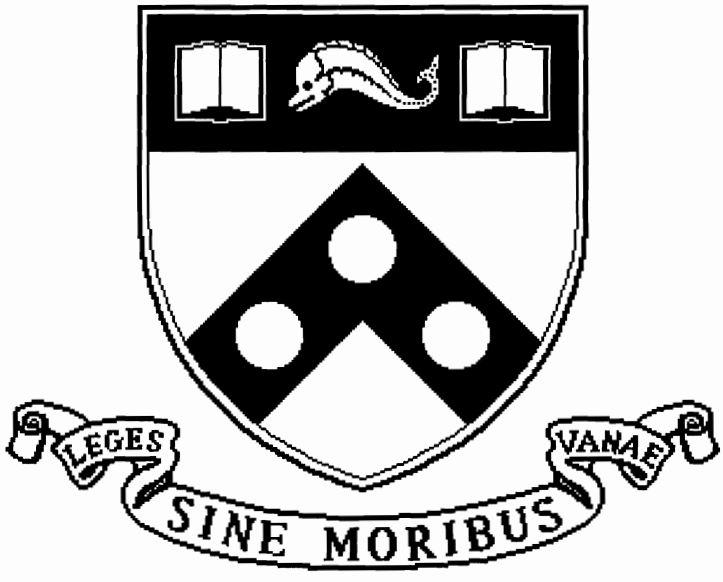
## Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-95-18.

# End-Point Resource Admission Control for Remote Control Multimedia Applications

MS-CIS-95-18

Klara Nahrstedt and Jonathan Smith




University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

April 1995

# End-Point Resource Admission Control for Remote Control Multimedia Applications

Klara Nahrstedt and Jonathan Smith*
University of Pennsylvania
e-mail: klara,jms@aurora.cis.upenn.edu

**Abstract**

One goal in certain classes of networked multimedia applications, such as full-feedback remote control, is to provide end-to-end guarantees. To achieve guarantees, all resources along the path(s) between the source(s) and sink(s) must be controlled. Resource availability is checked by the admission service during the call establishment phase. Current admission services control only network resources such as bandwidth and network delay. To provide end-to-end guarantees, the networked applications also need operating system resources and I/O devices at the end-points. All such resources must be included in a robust admission process. By integrating the end-point resources, we observed several dependencies which force changes in admission algorithms designed and implemented for control of a single resource.

We have designed and implemented the *multi-level admission service* within our *Omega architecture* which controls the availability of end-point resources needed in remote control multimedia applications such as telerobotics.

## 1 Problem Description

New applications enabled by multimedia devices involve the use of sensory data. These new applications become more interesting when distributed, but there are corresponding new research challenges. In particular, computer networks have traditionally been designed with resource-sharing goals in mind, e.g., the common Ethernet shared bus LAN. Emerging switched technologies such as Asynchronous Transfer Mode (ATM) offer the possibility of much greater control of the network subsystem's characteristics, expressed as Quality of Service (QoS) measures. The possibility of such control has inspired a rethinking of the architectures for application-to-application (end-to-end) communications in a distributed multimedia environment.

There is an expanding class of applications which, for a variety of reasons (such as insulating remote operators from hazardous materials, unavailability of a complex scientific instrument locally, etc.) require sensory feedback remote control. This class of applications shares many characteristics with teleoperation, which has been known for years (it was first looked at in the 1960s for space exploration tasks) to require hard real-time characteristics to preserve fundamental properties such

*This work was supported by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives. Additional support was provided by Bell Communications Research under Project DAWN, by an IBM Faculty Development Award, and by Hewlett-Packard.

as stability of the control system. We have abstracted the properties of such systems into what we call "Remote Control Multimedia Applications" (Figure 1) and use this abstraction to construct an architecture capable of supporting such applications. Among the most important algorithmic decisions to be made in such an architecture are those associated with the control of the time-sensitive system, such as admission and scheduling. We have developed new joint admission schemes for the type of complex systems (which are organized logically as multi-level systems) under study.

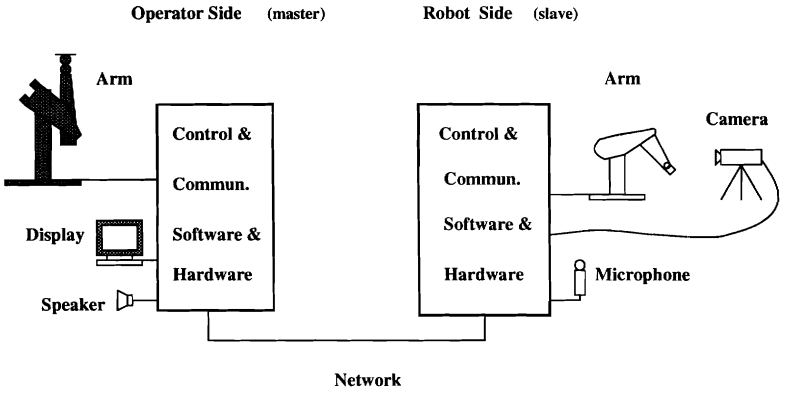



Figure 1: *A Remote Control Multimedia Application – Telerobotics*

The paper is organized as follows: Section 2 describes related work in this research area; Section 3 provides a brief overview of the Omega communication system in which our admission service is embedded; Section 4 describes the design of the multi-level admission service; Section 5 provides details on schedulability tests for *joint scheduling* of remote control multimedia applications; Section 6 gives an assessment of implementation issues which result from our *telerobotics* experiments; Section 7 concludes the paper.

## 2 Related Work

The majority of research on admission in distributed multimedia communication systems has focussed on *network resources*, such as bandwidth, network delay and buffer space for queues. Several admission mechanisms, scheduling policies and tests, and buffer allocation schemes are presented in [8, 10, 12] and other work.

Independent of network resources, CPU schedulability is analyzed in the real-time systems area. Several scheduling policies and tests are derived to control the availability of a single processor. For example, schedulability test for rate monotonic policy, currently preferred in multimedia systems, is derived in [16] and further analyzed in [14]. Deadline-monotonic scheduling for preemptive periodic and aperiodic tasks and its schedulability conditions are presented in [15, 13] and used for admission control of network tasks in [9]. Earliest deadline first algorithm is a dynamic algorithm which increases the processor utilization when scheduling aperiodic and period tasks [16]. Further, several real-time extensions for UNIX-compatible operating systems (OS) have been introduced, e.g., for Mach, AIX, Solaris and IRIX, to improve support of schedulability for 'delay sensitive' multimedia applications with their fixed priority preemptive scheduling (However, this scheduling remains inflexible in its purest form [2]).

The admission and control of individual resources has been achieved in several systems. However, there has been little orchestration between OS and network resources [9], and even less orchestration among all three types of resources (multimedia devices, OS resources, network resources). Yet, as we have shown through experiments [5], the availability of these resources is to a large degree interdependent. We discovered that these interdependencies exposed some serious limitations in the canonical real-time scheduling algorithms [16, 14, 15, 13], which we address in Section 5.

# 3 Omega Architecture

To specify admission service at the communication end-points, two issues need to be addressed: (1) the *communication system model* at the end-points, and (2) description of end-point resources.

## 3.1 Communication Model

The communication system is modeled as a two layer system (Figure 2). We call this end-point system architecture the *Omega Architecture.*

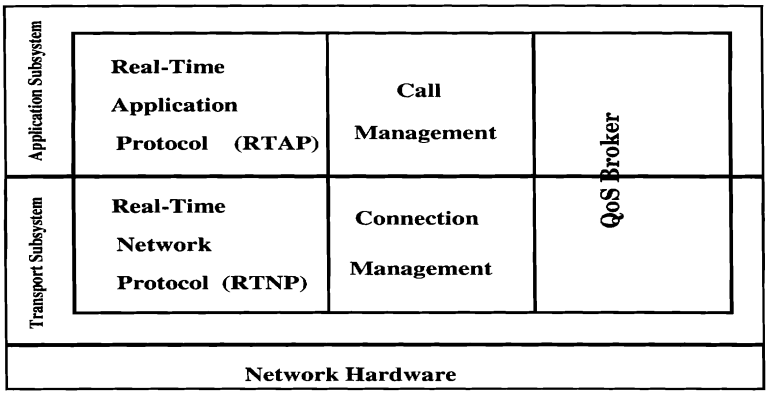



Figure 2: *The Omega Architecture Communication Model*

The *transport subsystem* layer includes the functionalities of the network and transport layers using *Integrated Layer Processing* [11]. Functions such as connection management, forward error correction, timing failure detection and timely data movement form the core of the *Real-Time Network Protocol (RTNP).*

The *application subsystem* layer contains the function of the application and session layers such as call management, rate control of multimedia devices, input/output functions (e.g., display of video), fragmentation of application protocol data units (APDUs), integration/disintegration of APDUs, etc. These functions are the core of the *Real-Time Application Protocol (RTAP).*

Both subsystems must provide a guaranteed transmission over specified calls/connections in application-to-application fashion. Therefore, they **require** guarantees on the resources needed for the communication. Resource guarantees are negotiated during the call establishment phase by the *QoS Broker* protocol [1] (Figure 3), which is an addition to the communication architecture. The broker orchestrates both local and global end-point resource availability. Local resource availability is achieved by using services such as *translation* (between application QoS and network QoS) and

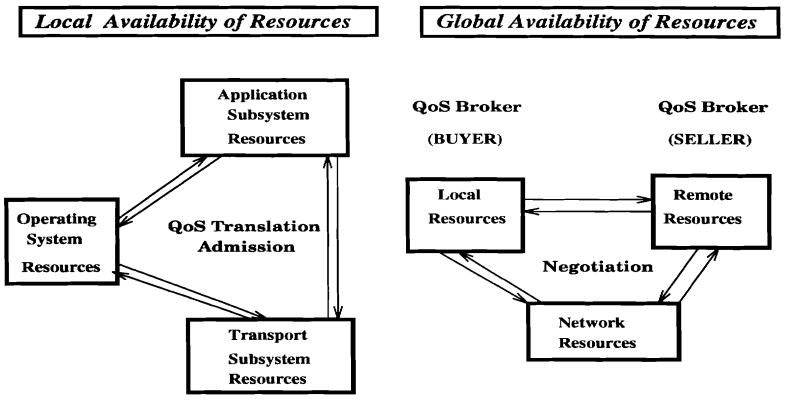



Figure 3: *The QoS Broker Concept*

*admission.* For global resource availability, the broker uses a *negotiation* service between the end-points and relies on network resource guarantees provided by the network subsystem, e.g., by B-ISDN switches. The goal of the broker is to negotiate a *resource deal* among all the system components (application, OS, network). The broker assumes different roles (*seller* and *buyer*) to distinguish between the participating partners.

### 3.2 Resource Model

At the end-point, three logical groups of resources must be managed, namely *multimedia devices*, *CPU scheduling and memory allocation* and *network resources.* We parameterize all end-point resources through *Quality of Service (QoS)* parameters maintained in small databases, which represent the requirements for the resources [1]. The resources in each domain (application, OS, network) maintain domain-specific representations. Therefore, we introduce *multiple views of QoS.* Thus, requirements of the application for multimedia devices are specified through *application QoS parameters.* For example, video quality is described through frame rate (30 frames/s), frame size (height * width in pixels), color (bits/pixel), etc. The *network QoS parameters* describe the requirements for the network resources, e.g., packet rate, packet loss, jitter, end-to-end delay. The *system QoS parameters* describe the requirements on CPU scheduling and buffer allocation (e.g., task start time , duration, and deadline).

To enforce coordinated management of the resources at the end-points, these multiple QoS views must be translated among each other. This is done by different services of the QoS Broker. For example, the translation between application and network QoS is done by the *QoS Translator* [6]. These different QoS representations are also used by the multi-level admission service, described in the next section.

## 4 Admission Control

Admission control is an essential element to achieve guaranteed services. For distributed multimedia communications systems, each resource must keep track of its availability along the path(s) between source(s) and sink(s). The state diagram of an admission service is shown in Figure 4.

As we stated in the previous section, three groups of resources are managed at the end-points.

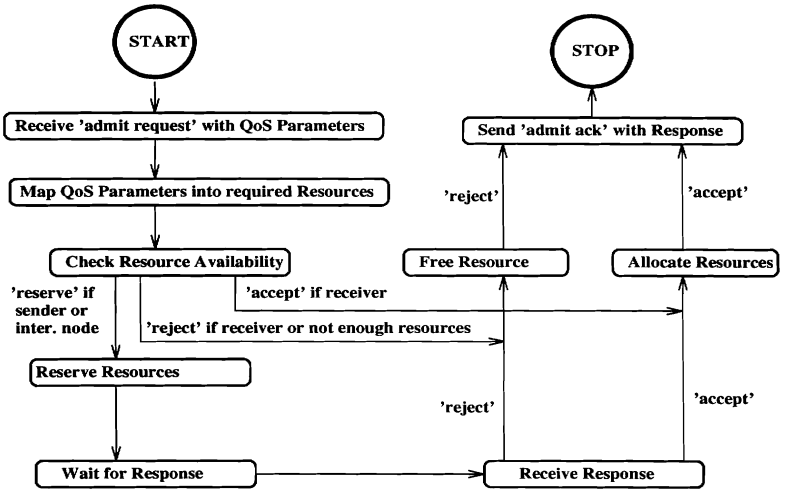



Figure 4: *State Diagram of an Admission Service per Resource*

This implies that the admission service must be performed at several levels. We split the admission service into two levels according to the communication subsystem layering (Figure 2). Hence, the QoS broker protocol uses the *admission service* at the application and transport subsystem levels.

For convenience, we assume networked multimedia applications with periodic media streams (e.g., uncompressed video, sensory data). Our admission tests are therefore limited to providing guarantees for this type of traffic. Aperiodic requests (tasks) may occur (e.g., QoS renegotiation/resource adaptation request), however for these requests our scheduler polls periodically and treats them as deadline-driven requests (tasks).

## 4.1 Admission Service in the Application Subsystem

The admission service performs four tests at the application subsystem level: (1) device quality test, (2) local schedulability test, (3) end-to-end delay test and (4) buffer allocation test. These tests check the multimedia devices and system resources availability for the real-time networked application.

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 as 30 frames/second, then the admission must reject the QoS requirement.

The *local schedulability test* takes the system QoS parameters which specify the application tasks for processing of multimedia streams (task duration, task period, task deadline, dependency relations) and checks if the tasks are schedulable. We discuss scheduling policies and tests for this level in the next section.

The *end-to-end delay (EED) test* takes the duration of the application tasks and checks them against the specified QoS EED bound. Here, we make sure that the tasks, although schedulable, don't violate the EED requirement. This is especially important in cases where *task period* $> EED$. For example, sensory data in telerobotics provide such a behavior (e.g., the task period is 20 ms and EED 10 ms).

The *buffer allocation test* checks if there is enough memory space for the ring buffers assigned

to multimedia devices to smooth the traffic jitter. This is necessary when *measured EED* < *requested EED*. Real-time networked applications want the right data at the right time (requested EED), not sooner or later (although sooner is still better than later).

### 4.2 Admission Service in the Transport Subsystem

The admission service at the transport subsystem level performs tests on network resources such as a throughput test, rate control test, network EED test, and system resources such as schedulability test.

The *throughput test* controls the assignment of bandwidth to individual connections. The upper bound of available aggregate throughput at the end-point is determined by the network host interface and its device driver. For example, in our system the ATM host interface (hardware) provides a transmission rate of 155 Mbps, however, the ATM transport subsystem, after overhead, provides 135 Mbps [7]. Hence, any throughput requested for the sending or receiving connections is checked against the 135 Mbps limit 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 (in our implementation, 1000). This bound results from the OS cost (due to overhead) of moving network packets between the user and kernel space.

The *end-to-end delay test* checks the duration of all tasks (application and network tasks) at the end-points against the required end-to-end delay bound.

The *schedulability test* checks the schedulability of all tasks (application and network tasks).

The *buffer allocation test* is needed if the network tasks queue the incoming/outgoing packets. Our current system queues packets (ATM cells) in the network host interface (ATM layer) and application PDUs at the application subsystem level, but not in the transport subsystem.

## 5 Schedulability

For the *schedulability test*, the parameters of interest are: (1) task duration, $e$; (2) task period, $P$; and (3) context-switch time between two OS processes/threads, $cs$.

Further, we assume that all tasks (application and network) are *non-preemptive* basic tasks (e.g., *read sensory sample*, *read a video frame* from a video device). The reason is that although many multimedia communication systems, when testing for schedulability, assume preemptive scheduling algorithms, these algorithms assume that any message can be suspended at any time, with a small overhead, in order to transmit a higher priority message. However, in communication systems, a high preemption rate is usually synonymous with high message overhead. To avoid this message overhead, we adapt a non-preemptive scheduling algorithm. Non-preemptive algorithms are relatively easy to implement, but the drawback is that a high priority message can be blocked by a long low priority message. This is called *priority inversion* [3]. To avoid this effect (at least at the processor level), we negotiate (admit) the proper size of the long low priority message (e.g., proper size of the fragment for uncompressed video frame) during the brokerage phase.

## 5.1 Schedulability Test in the Application Subsystem

At the *application subsystem level* tasks have periodic behavior and provide read and write operations from/to the multimedia devices. There are also some aperiodic tasks, such as requests for renegotiation, which have a deadline-driven behavior. Therefore, we can test these tasks as if we scheduled using the *earliest deadline first (EDF)* policy. For this kind of scheduling, Liu and Layland provide a schedulability test in [16]. However, because our tasks are non-preemptive, the schedulability test must be altered [1].

Let $e^A_{o,i,r}$ specify the duration of an application $A$ task $r$ for medium $i$ (e.g., video/sensory data) sample in direction $o$ (input/output). Let $cs^A_j$ be the *j-th* context switching time between application $A$ tasks. Let $min(P_{i,o})$ represent the minimal period among the media $i$ sample periods $P_i$ (inverse of sample rate) in direction $o$. The schedulability test in the application subsystem is:

$$T^A = \sum_o \sum_i \sum_r e^A_{o,i,r} + \sum_j cs_j \leq min_{o,i}(P_{o,i}) \tag{1}$$

Further, for each medium $i$ in direction $o$, the following must hold:

$$\sum_r e^A_{o,i,r} \leq P_{o,i} \tag{2}$$

If the schedulability test (1) cannot be met, the stream with later deadline (lower rate) will be rejected. If the schedulability test is satisfied, the task priorities are assigned according to their deadline (highest priority is assigned to the earliest deadline). If there are input/output tasks with the same period, the input tasks get higher priority than the output tasks.

## 5.2 Schedulability Tests in the Transport Subsystem

For the transport subsystem, let $e^{NET}_{o,k,r}$ denote the processing time of the task $r$ performed over connection $k$ packet in direction $o$ in transport subsystem *NET*. Depending on the implementation of network tasks, $cs^{NET}_n$ represents the *n-th* context switch between network tasks.

The scheduling at the transport subsystem level, where we test schedulability of tasks (application and network tasks) sharing a single processor, must consider the following time dependencies:

1. *Time dependencies between application and network tasks*

   We can't use the EDF and priority assignment as discussed in Section 5.1. at the transport subsystem level. We show this with a counter example.

   Consider an application task which reads video frames from a video device. Network tasks send packets (fragments of a video frame, which are typically larger than the largest network packets) (Figure 5). If we assume EDF policy for this example, we assume that the application and network tasks are independent, periodic, their deadlines are task periods and the priorities are assigned according to their deadlines. In our example (Figure 5), the network tasks have earlier deadline than the application task, therefore they would be scheduled first which is semantically wrong (network tasks can't send packets which don't exist).

   The application and network tasks share a single processor, are time dependent on each other, and network tasks may not be strongly periodic, as is the case for application tasks which

[1] The schedulability test is tighter for non-preemptive tasks: $\sum_{i=1}^{n} \frac{e_i}{P_i} \leq \frac{1}{min_i(P_i)} \sum_{i=1}^{n} e_i \leq 1$

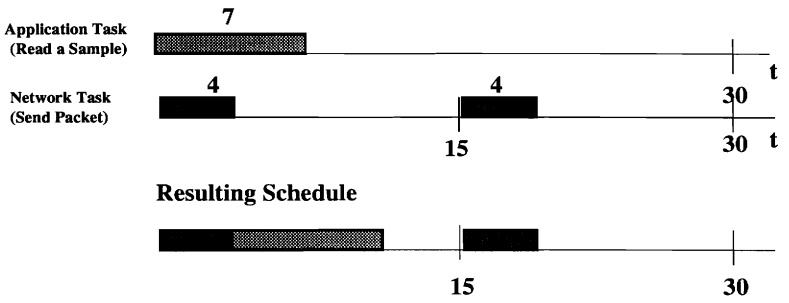



Figure 5: *Counter Example*

must be considered in the schedulability tests and priority assignments. In our example, the *dependency (precedence* $\longrightarrow$ *[4])* relation is $read\_sample(k) \longrightarrow send\_packet(k_1) \longrightarrow send\_packet(k_2)$. A further implicit precedence between application and network tasks is $receive\_packet(k) \longrightarrow write\_sample(k)$.

The priority is assigned by the application subsystem to the application tasks (according to the deadline) and the network tasks must *inherit* these priorities in order to enforce *joint scheduling.*

The schedulability test in the transport subsystem for this type of dependency is:

$$T^A + \sum_o \sum_k \sum_r e^{NET}_{o,k,r} + \sum_n cs^{NET}_n \leq min(P_{o,i}) \tag{3}$$

$$\sum_r e^{NET}_{o,k,r} \leq P_{o,k} \tag{4}$$

The added network tasks (sending/receiving) might violate the schedulability test, hence, they can be preempted to the next interval (in the case of sending tasks) if they satisfy the network EED test [2]. In the case of receiving tasks, the application task might be preempted (see Figure 8). Again, the EED test needs to be checked too. Hence, the schedulability test, especially the decision of preemption of tasks to the next intervals, is coupled to the end-to-end delay test.

2. *Time dependencies between input/output streams*

   When testing for schedulability of tasks at the end-points, other types of time dependencies can occur and must be considered.

   For example, Figure 6 shows sensory data dependency relations in our telerobotics application, where the operator sends position data $m_k$, the slave receives them and sends back force feedback data $f(m_k)$. The application would like to receive $f(m_k)$ so that the computation of sample $m_{k+1}$ can be based on $f(m_k)$ $(write(f(m_k)) \longrightarrow read(m_{k+1}))$ .

   If this kind of dependency occurs, a *wait for feedback* time interval must be included into the schedulability test because the input and output stream information are interdependent.

[2] Number of possible intervals to preempt a task is $\frac{lcm(P_1,\ldots,P_n)}{min_i(P_i)}$

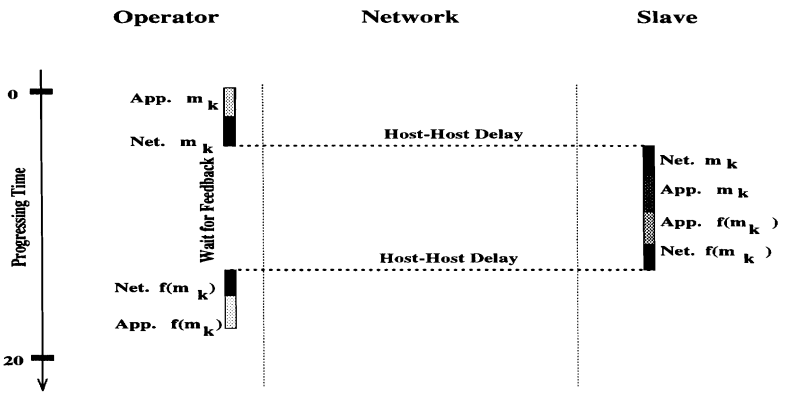



Figure 6: *Distributed Scheduling - Precedence Graph (Example)*

The schedulability test for these types of dependencies at an end-point (e.g., the operator side in the telerobotics) is:

$$T^A + \sum_o \sum_k \sum_r e^{NET}_{o,k,r} + \sum_n cs^{NET}_n + WFF \leq min(P_{o,i}) \tag{5}$$

where WFF (Wait for Feedback) is specified as:

$$WFF = 2 \times HHD + \sum_{k(i)} \sum_r e^{NET}_{in,r,k} + \sum_r e^{A}_{in,r,i} + \sum_{k(i)} \sum_r e^{NET}_{out,r,k} + \sum_r e^{A}_{out,r,i} \tag{6}$$

HHD is the host to host-interface delay. The knowledge of WFF time can be utilized for scheduling of another task which serves a different medium. At the slave side the schedulability test (3) can be used.

The QoS broker gets the application precedence relations from the user (through application QoS parameters) and together with the implicit application/network precedence relations it creates a *precedence graph.* According to the precedence graph, negotiation and admission services provide the distribution and acceptance of the system QoS parameters (tasks). The broker suggests a joint scheduler based on *time slicing* (slicing feasibility and a solution to the slicing problem are described in [4]).

# 6 Implementation Issues

The admission service is executed at each point along the path between source(s) and sink(s). The establishment of a resource deal between operator and slave sides by the QoS Broker and the placement of the admission service in the protocol are shown in Figure 7.

The admission service has access to profiles, which store the application QoS in an application profile (these are the databases we mentioned earlier in Section 3.2.), the system QoS in the system profile and the network QoS in the network profile. When all resources are allocated, the brokered deal for each group of resources is added to these profiles.

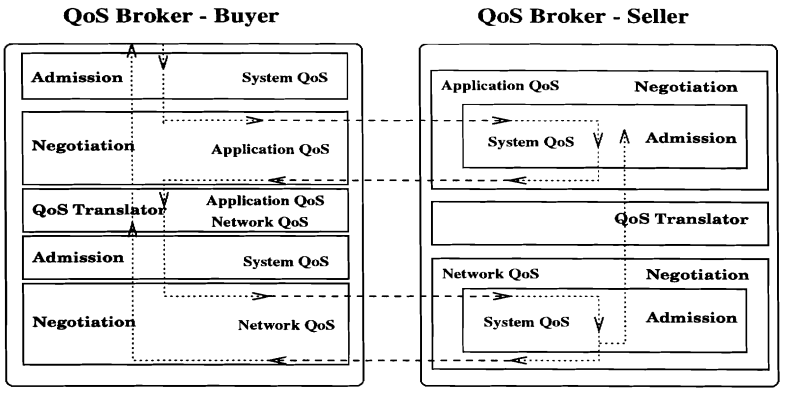



Figure 7: *Establishment of a Resource Deal - QoS Broker Protocol*

One issue with respect to the system profile must be mentioned. The system profile at the beginning includes *a priori* precomputed task durations for each medium supported at the end-point which participates in the real-time networked multimedia application. This is required to make the schedulability decisions. The result of the schedulability tests is a suggested feasible schedule of all the tasks participating in that particular application. This schedule is stored in the system profile as the deal for CPU scheduling. An important part of computing a feasible schedule is to determine the *least common multiple (lcm)* among all the I/O media periods, so we known how many intervals ( $\frac{lcm_{o,i}(P_{o,i})}{min(P_{o,i})}$) might be scheduled differently. An example of a joint schedule at the operator side for transmitting (1) one sensory stream from operator to slave (application task period - 20 time units), (2) one sensory stream from slave to operator (application task period - 20 time units) and (3) one video stream from the slave to the operator (application task period - 60 time units) is shown in Figure 8. The *lcm* is 60 time units, and the number of intervals, scheduled differently, is 3.

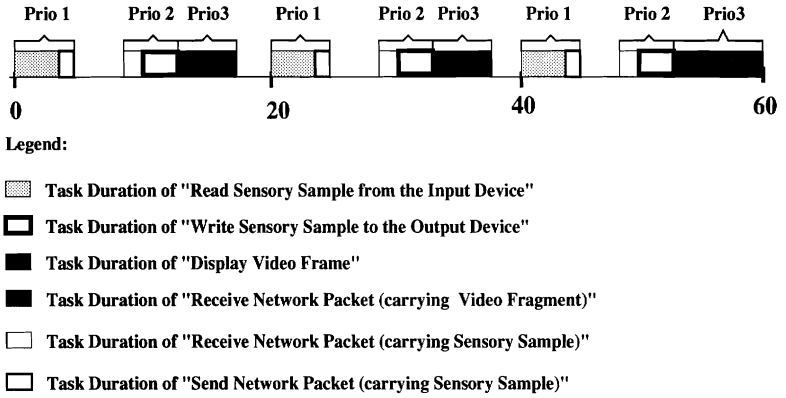



Figure 8: *Example of a Joint Schedule*

A prototype of the Omega architecture is currently implemented on the IBM RS/6000 workstation where we utilize the real-time (RT) extension support (RT priorities with fixed-priority scheduling, and a page locking mechanism) of the AIX OS. However, the AIX RT extension does not provide enough control of the AIX scheduler to the user, therefore we split the scheduling. The

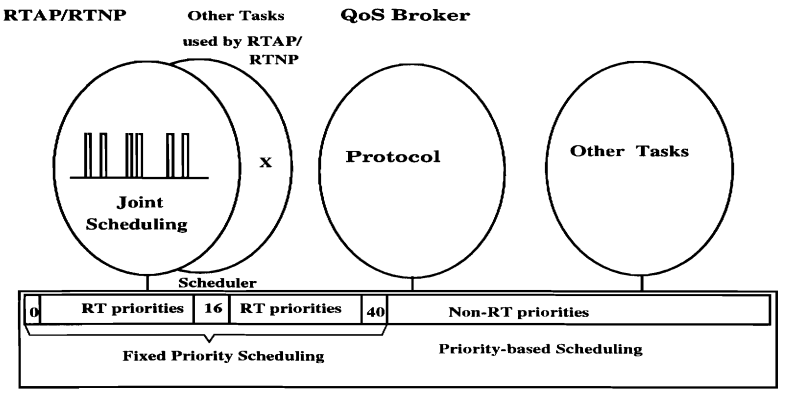



Figure 9: *Mapping of the Scheduling*

networked application and network tasks (RTAP/RTNP) run as a separate process where the individual tasks are scheduled according to the joint scheduler, and the single process uses fixed priority scheduling (Figure 9). We assign to the process(es), which need RT guarantees, RT priorities higher than the AIX scheduler. This guarantees that the process is not preempted by the scheduler.

The RTAP/RTNP tasks perform very well under joint scheduling as implemented. The measured end-to-end delays of the sensory data for our telerobotics application are 2 ms (average value) using an ATM LAN environment [1], which is a factor of 500 better than the application had previously achieved with TCP/IP over Ethernet (1.2 sec!).

However, when several applications share the processor, and the additional applications don't register with the QoS broker, and hence don't undergo admission control of the CPU scheduling, the AIX RT extension cannot provide guarantees. Thus once another process is scheduled (even a non-real-time process), priority inversion may occur, and the real-time task under joint scheduling misses the deadline.

# 7 Conclusion

Admission service is an important element for prediction of resource availability when end-to-end guarantees are required from the communication system. The communication system tasks require not only network resources but also multimedia devices and system resources at the end-points. Current admission services for single resources must be extended. Our multi-level admission service provides this extension. Our studies of admission among the end-point resources showed several time dependencies which must be included into schedulability tests to provide correct CPU scheduling.

We validated our admission service by using it in the implementation of a non-trivial telerobotics application. In this application, we successfully provided hard-real time guarantees for the sensory data and soft-real-time guarantees for the video traffic.

# References

[1] K. Nahrstedt, J. Smith, "The QoS Broker", *IEEE Multimedia*, Spring 1995, pp.53-67

[2] N.C. Audsley, R.I.Davis, A. Burns, "Mechanisms for Enhancing the Flexibility and Utility of Hard Real-Time Systems", *Proceedings Real-Time Systems Symposium*, San Juan, Puerto Rico, December 7-9, 1994

[3] R. L.R. Carmo et al. "Real-Time Communication Services in a DQDB Network", *Proceedings Real-Time Systems Symposium*, San Juan, Puerto Rico, pp. 249-258, December 7-9, 1994

[4] M. Di Natale, J. A. Stankovic, "Dynamic End-to-end Guarantees in Distributed Real-Time Systems", *Proceedings Real-Time Systems Symposium*, San Juan, Puerto Rico, pp. 216-227, December 7-9, 1994

[5] K. Nahrstedt, J. Smith, "Experimental Study of End-to-End QoS", MS-CIS-94-08, Technical Report, University of Pennsylvania, February 1994

[6] K. Nahrstedt, J. Smith, "A Service Kernel for Multimedia Endstations", *Multimedia: Advanced Teleservices and High-Speed Communication Architectures*, 1994, Heidelberg, pp. 8-22

[7] J. Smith, B. Traw, "Giving Applications Access to Gb/s Networking", *IEEE Network*, July 1993

[8] J. M. Hyman, A. A. Lazar, G. Pacifici, "A Separation Principle Between Scheduling and Admission Control for Broadband Switching", *IEEE JSAC*, Vol. 11, No. 4, May 1993, pp. 605-616

[9] H. Tokuda, Y. Tobe, S.T.-C. Chou, J. M. F. Moura "Continuous Media Communication with Dynamic QoS Control Using ARTS with an FDDI Network",*Proceeding SIGCOMM '92*, Baltimore, ML, pp. 88-98, August 1992

[10] T. Murase, H. Suzuki, S. Sato, T. Takeuchi "A Call Admission Control Scheme for ATM Networks Using a Simple Quality Estimate", *IEEE JSAC*, Vol. 9, No. 9, December 1991, pp. 1452-1460

[11] D.D. Clark, D.L. Tennenhouse, "Architectural Considerations for a new generation of protocols", *Proceedings ACM SIGCOMM '90*, Philadelphia, PA, pp. 200-208, September 1990

[12] D. Ferrari, D.C. Verma, " A Scheme for Real-Time Channel Establishment in Wide-Area Network", *IEEE JSAC*, Vol. 8, No. 3, April 1990, pp. 368-379

[13] N. Audsley, "Deadline Monotonic Scheduling", *Technical Report*, Department of Computer Science, University of York, Heslington, York, September 1990

[14] J. Lehoczky, L. Sha, Y. Ding, "The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", *Proceedings IEEE Real-Time System Symposium*, Santa Monica, California, pp. 166-171, IEEE Computer Society Press, 5-7 December 1989

[15] J. Y. T. Leung, J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks", *Performance Evaluation (Netherland)*, 2(4), pp. 237-250, December 1982

[16] C. L. Liu, J. W. Layland, "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, 20(1), pp. 40-61, 1973