A Service Kernel for Multimedia Endstations

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

Abstract

Quality of Service (QoS) guarantees for *delay sensitive* networked multimedia applications, such as teleoperation, must be application-to-application. We describe a set of services, a *service kernel*, required at the end points, for multimedia call establishment with QoS guarantees. These services provide: (1) *Translation* among different domain specifications (layer-to-layer translation) and domain-resource specifications (layer-to-resource translation); (2) *Admission and Allocation* of resources; and (3) *Negotiation and Coordination* of QoS specifications among the distributed end-points. For each service we propose architectural solutions.

We are testing the service kernel with an ATM-based telerobotics application.

1 Problem Description

Quality of Service (QoS) guarantees for delay sensitive networked multimedia applications must be application-to-application. Guarantees are achieved if: (1) the information is carried between end-points using delay-bounded communication protocols [6], [5], (2) the end-points use delay-bounded services of the operating system (OS) [4], [9], and (3) the application, OS and network are able to prepare and configure the environment for delay sensitive multimedia calls with QoS guarantees.

We identify a set of services required for QoS guarantees(in end-to-end multimedia establishment protocols) in a multimedia environment at the end-points. We call this set a *kernel*, because while additional services may be required (e.g., services for establishment of a video conference), these particular service are essential.

[*]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.

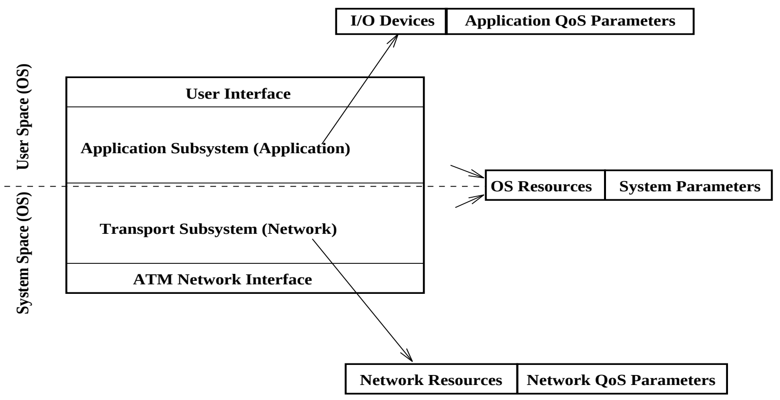

Figure 1: *End-Point Model with Application, Transport Subsystems and Operating System Specifications*

We believe that kernel services must provide the following functions: (1) *Admission and Allocation* of resources (e.g., task scheduling) for the local processor; (2) *Admission and Allocation* of network resources, such as bandwidth and pacing requirements; (3) *Negotiation and Coordination* among the other application end-points; and (4) *Translation* among different resource and domain (application, OS, network) specifications. For each, we propose architectural solutions.

2 End-Point Architecture

The kernel services assume the end-point model of Figure 1. The communication stack consists of an *user interface*, an *application subsystem*, a *transport subsystem*, and a *network interface*. Both subsystems are embedded in a multi-user/multi-process OS environment.

An application identifies its specific requirements to the application subsystem using *application QoS parameters* (Figure 2). The application QoS parameter structure describes a multimedia stream in one direction. Hence, one has to keep in mind that for both directions (input and output) a multimedia stream description has to be given. The media quality component consists of an *interframe* specification and an *intraframe* specification. The interframe specification gives the characteristics of the homogeneous medium stream. If the individual samples in the stream differ in quality, intraframe specification has to occur. The parameters are stored in an application database.

The transport subsystem is configured with *network QoS parameters* (Figure 3), which describe the requirements on the quality of the network connection. The network QoS parameter structure (Figure 3) describes the QoS of data which is transmitted over one network connection. The network QoS parameters are stored in a network database at the end-point. Hence, the network database

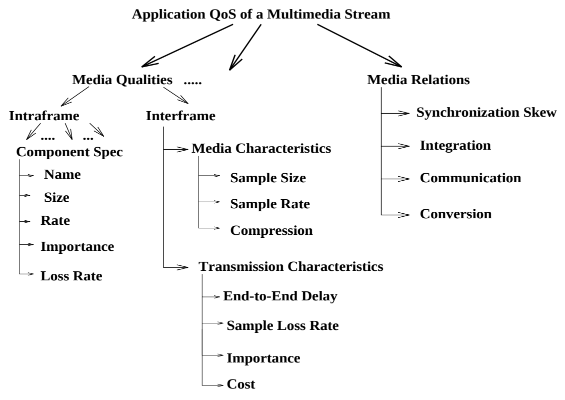

Figure 2: *Application QoS Parameters*

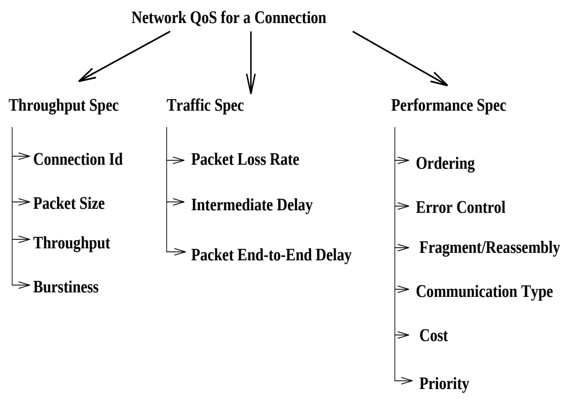

Figure 3: *Network QoS Parameters for a Connection*

includes as many network QoS descriptions as there are active connections for sending and receiving data.

The OS behavior is specified by the *system parameters* (Figure 4) which are stored in a system database. The system parameters mirror the requirements of the multimedia stream for the OS resources across both subsystems. They consist of *Task Specifications* for each medium and each connection, *Task Scheduler Specifications* and *Space Requirements.* The duration times of tasks are precomputed. The system database includes these parameters for both directions.

Based on this end-point model, the following three service groups are needed to support call/connection establishment with QoS guarantees:

- *Translation Services* provide mapping of parameters between communication layers (application QoS and network QoS) as well as between layer and resource (application/network QoS and system QoS). The functionalities of these services are given in Section 3.

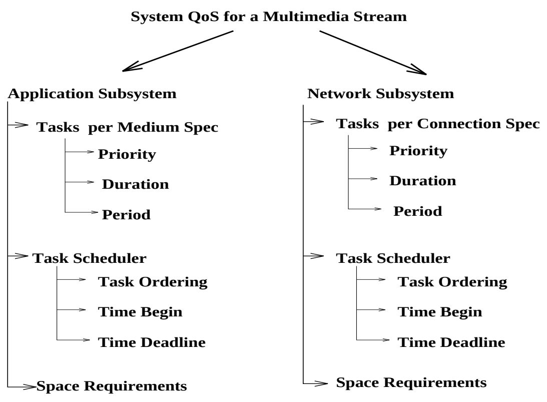

Figure 4: *System QoS for a Multimedia Stream*

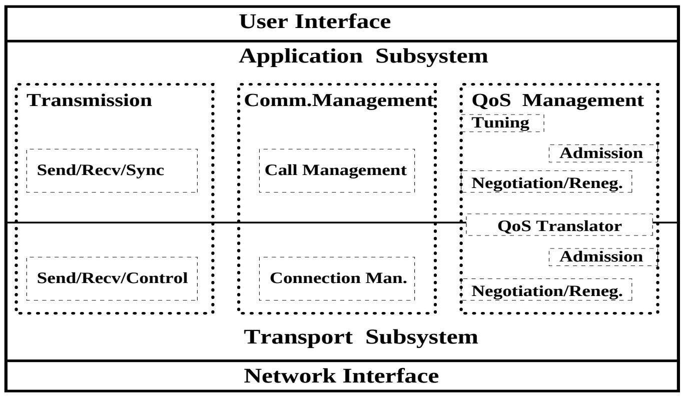

Figure 5: *End-Point Communication Architecture*

- *Resource Admission Services* provide a control mechanism for shared resource availability. Discussion of these services is presented in Section 4.
- *Information Distribution Services* implement layer-to-layer and peer-to-peer communication for call establishment with QoS guarantees. A brief discussion is presented in Section 5.

The translation, admission, and distribution services are added to call/connection QoS management. If an application requires guaranteed services, one or more of these services will be invoked. If no guarantees are required, traditional 'best-effort' call/connection establishment management is used. The resulting end-point communication architecture is shown in Figure 5. We are testing this end-point architecture and its QoS service kernel with an ATM-based telerobotics application. Its hardware setup and implementation are described in Section 6.

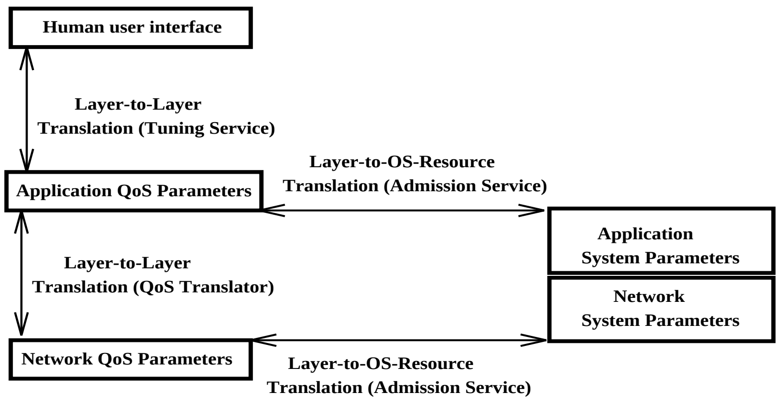
Human user interface
Layer-to-Layer Translation (Tuning Service)
Application QoS Parameters
Layer-to-OS-Resource Translation (Admission Service)
Application System Parameters
Layer-to-Layer Translation (QoS Translator)
Network System Parameters
Network QoS Parameters
Layer-to-OS-Resource Translation (Admission Service)

Figure 6: *Relations among the Translations*

3 Translation Services

Our description of the end-point implies that parameter sets have to be mapped onto one another. Mapping between communication layers leads to *layer-to-layer translation* functionality; mapping between layers and their corresponding system resources requires a *layer-to-resource translation* functionality. The relation among the translations is shown in Figure 6.

3.1 Layer-to-Layer Translation

The layering in Figure 1 indicates that we need layer-to-layer translation between the subsystems. Specific instances of such translation have been noted, for example, between ATM Adaptation Layer (AAL) and ATM Layer [8].

There are other layer interfaces where translation needs to occur. An important property of these translations, such as the one between application and transport subsystem, is *bidirectional translation*, which may cause problems of ambiguity. For example, in the case of video transmission, if bandwidth cannot be allocated for a video stream, the translation from the transport subsystem to the application subsystem may result in either a decrease of frame resolution quality, a decrease of frame rate, or both.

In our communication architecture (see Figure 5), the translation between the user and application is performed by the *tuning service*. This service maps the application QoS parameters onto audio/visual descriptions and vice versa. In our current implementation, a motion video file is used for visualization of the frame rate parameter. An important issue here is that the frame rate tuning must be based on the balance between the network capabilities and end-point OS capabilities[1] [2].

[1]Locally, the tuning service can display, for example, 15 frames per second, but when a network is involved between the video source and the destination, the frame rate may not be sustainable. The frame rate may decrease due to fragmentation/reassembly at the transport

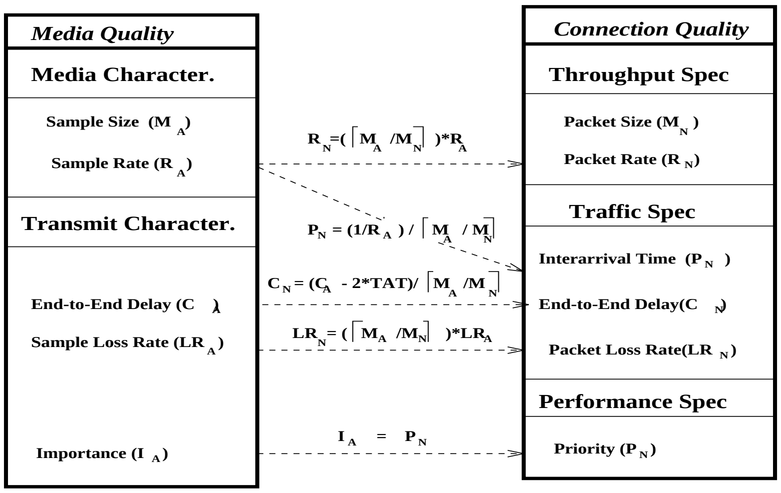

Figure 7: *One-to-One QoS Translation*

The translation between the application and the transport subsystem is performed by the *QoS Translator* entity. The QoS Translator maps application QoS parameters onto network QoS parameters and vice versa. The translation includes at least three activities:

1. *One-to-one translation* involves a translation between the network connection quality and the medium quality. Figure 7 shows some cases of these transformations[2]. Other information transformations are:

   - *Media Relations* structure (Figure 2) in the application QoS provides the specification of *communication* (unicast/multicast/broadcast). This information gets copied to the *communication type* in the specification of the network QoS (Figure 3).
   - *Throughput* is computed from the *packet size* (M_N) and *packet rate* (R_N): $Throughput = R_N \times M_N$. This computation occurs after the translation from R_A to R_N (Figure 7).

     An ambiguity case, because of the bidirectional translation, may arise in cases when the network cannot guarantee the requested throughput and it suggests a lower throughput to the QoS Translator. The implications are as follows: The change in throughput influences first the packet rate, R_N. We assume that the packet size is fixed. The

subsystem. Therefore, one has to be mindful of what the tuning service promises and what the multimedia communication system can deliver.

[2]We specify single value (average) deterministic application QoS parameters, therefore the translated parameters (e.g., interarrival time P) are also single value deterministic parameters. We assume that the end-to-end delay C_N depends on TAT, which is processing time of application tasks to process a sample in the application subsystem. For loss rate LR_N, the transformation holds only for case where a sample and packet(s) are correlated.

changed packet rate, R'_N, influences sample rate, R_A, and sample size (M_A) as follows:

$$M_A = \lfloor \frac{R'_N}{R_A} \rfloor \times M_N$$

$$R_A = \frac{R'_N}{\lceil \frac{M_A}{M_N} \rceil}$$

- *Fragmentation* is set TRUE when sample size is bigger than the packet size. If fragmentation occurs, it influences the computation of the end-to-end delay (C_N) for the packet as shown in Figure 6.
- *Ordering* is set TRUE, if continuous media with real-time behavior are sent. In non-real-time media behavior, the ordering requirement merely depends on the application's ability to handle out of order data.
- *Error Control* depends on the importance parameter and sample loss rate of the medium quality. If real-time behavior of the continuous media is required, its importance is high and sample loss rate is low, then a *forward error correction (FEC)* [7] mechanism is used in the communication protocol.[3] Otherwise, a different error correction mechanism (e.g., retransmission) can be specified, if supported by the communication protocol suite.
- *Cost* and *Burstiness* mappings are currently not supported.

2. *Integration* means interleaving (multiplexing) different media into one media stream which will be sent through one network connection. This implies that the different media qualities have to be merged into a new medium quality specification (many-to-one), as shown in Figure 8. After integration of the media qualities, one-to-one translation occurs between the resultant medium quality and the network QoS for the connection. It is important to point out that the resultant medium quality is the union with precedence of the media qualities being integrated. Therefore, integration should be done on media which have similar QoS requirements.

   Because the translation is bidirectional, ambiguities can also occur in this case. Therefore, the QoS Translator passes to the application several possibilities and lets the user decide which medium will suffer in quality. In a more sophisticated system, a rule-based QoS Translator can be deployed which will make decisions based on rules given by the user *a priori*.

3. *Disintegration* means splitting of a medium stream into several streams which will be transmitted through several connections. This case occurs

[3]We use our own *RTNP – Real-Time Network Protocol*. It is a rate-based network protocol working above the AAL 4 layer. RTNP currently supports FEC if high reliability is required. Otherwise error correction is turned off. RTNP has error-detection and it reports to *RTAP (Real-Time Application Protocol)* about missing information. RTAP must resolve the conflict when missing information occurs.

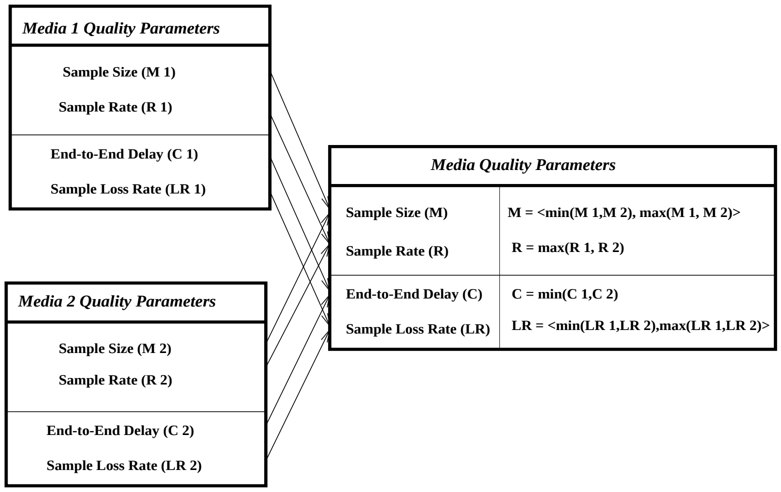

Figure 8: *QoS Integration*

when the medium stream carries different kinds of information (e.g., in a MPEG compressed video stream we have specification of I-frames, P-frames, and B-frames). Since the interframe medium quality specification includes the intraframe specification, the QoS Translator can perform one-to-one translation immediately between the intraframe component specification and the network QoS.

As an example, the architecture of the QoS translator process for a robotics stream, an audio stream, and an uncompressed video stream is shown in Figure 9. [4]

3.2 Layer-to-OS-Resources Translation

Each communication layer uses OS resources; hence, a mapping between the layer QoS parameters and system parameters is required. We consider translations between application QoS parameters and OS resources with respect to the application subsystem protocol (RTAP), as well as network QoS parameters and OS resources with respect to the transport subsystem protocol (RTNP). This mapping is done within the *admission services*. The application QoS parameters are mapped onto the system parameters (Figure 4): (1) task priorities, (2) task periods, and (3) buffer space requirements. Likewise, the network QoS parameters must be mapped onto the system parameters (Figure 4).

The *task priorities* are inherited from the importance of the medium quality and connection priority. The importance of priority inheritance for support of guarantees is experimentally shown in [2].

[4]In our telerobotics application, split of the sensor data occurs because the robotics stream carries 4 components (N, O, A, P), which have different meaning and importance for the movement of the robot arm.

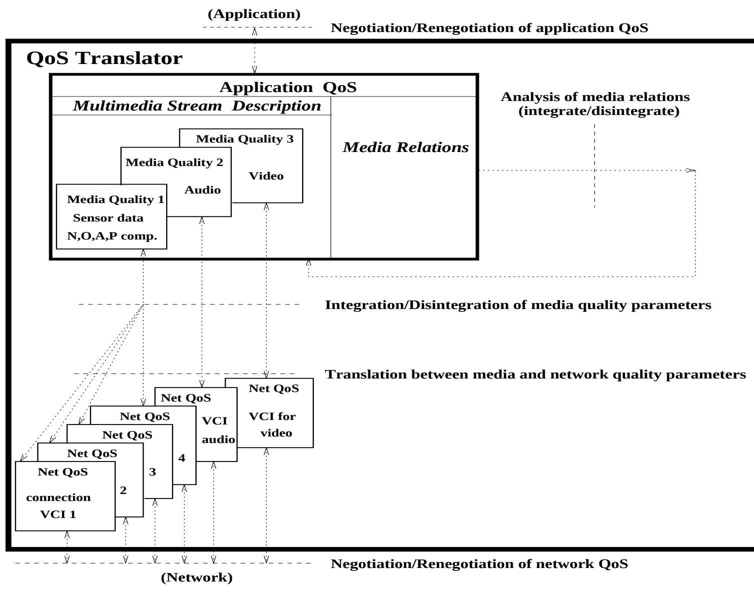
(Application)
Negotiation/Renegotiation of application QoS
QoS Translator
Application QoS
Multimedia Stream Description
Media Quality 3
Media Quality 2
Video
Audio
Media Quality 1
Sensor data
N,O,A,P comp.
Media Relations
Analysis of media relations
(integrate/disintegrate)
Integration/Disintegration of media quality parameters
Translation between media and network quality parameters
Net QoS
VCI for
video
Net QoS
VCI
audio
Net QoS
4
Net QoS
3
Net QoS
2
Net QoS
connection
VCI 1
Negotiation/Renegotiation of network QoS
(Network)

Figure 9: *The QoS Translator (Example)*

Task durations as well as specification of tasks in communication protocols are pre-computed and stored *a priori* in the system database. This parameter depends on the sample (packet size).

Task period is computed as the inverse of the sample rate (packet rate).

Task schedulers are computed from the the rates and priorities of the tasks in both subsystems (we use a mixed scheduling algorithm [11] which is composed of a rate-monotonic and deadline-driven algorithm). The important issue here is that the task scheduler in the network subsystem takes into account the task scheduler in the application subsystem. The reason is that the end-point has one processor Therefore, both task schedulers have to be in balance if guarantees are to be achieved.

The sample size (packet size), the fragmentation/reassembly, the integration, the error control, and disintegration determine the *space requirements* in the system QoS (both subsystems). In our communication protocols, there must be allocated for each direction at least $2 \times S_A$ space. The reason is that the application subsystem and the transport subsystem do not share the space where the sample is loaded. Therefore, at least one 'copy' operation occurs between the subsystems. Fragmentation/reassembly, FEC, and disintegration introduce an increase of space requirements at the transport subsystem. Integration can result in decrease of space requirements at the transport subsystem, but also may not.

4 Resource Admission Services

For call establishment with QoS guarantees, *shared resource availability* must be checked. The control is performed by the resource admission services.

There are differences between resource admission services in network switches and end-points: An admission service in the network switches limits its responsibility to shared network resources. Thus, the resource admission service relies only on network resource availability tests (e.g., a schedulability test for admission of outgoing traffic over a shared link.) Also, in the case of ATM networks there is only the ATM layer, so all admission tests are at the cell level.

End-point admission is more complex. The OS resources (e.g., CPU) as well as network resources (e.g., bandwidth to/from the network interface) are shared. Further, we have several communication layers; therefore, we have multilayer admission with multiple resource availability tests, such as *OS resource availability tests* and *network resource availability tests*. In our end-point model, we assume admission services in the application subsystem and in the transport subsystem. Further, these admission services rely on the resource admission service in the ATM network.[5]

4.1 Assumptions for Admission Tests

For specification of schedulability tests (and a feasible task scheduler) in the admission services, it is important to specify:

1. *Types of Tasks*

   We examine applications with (i) *periodic tasks*, due to periodic production of media; and (ii) *deadline-driven tasks* for the movement of data from/to devices.[6] The deadline-driven tasks can be further classified into *hard-real-time deadline* tasks which process media streams such as tactile and kinesthetic data; *soft-real-time deadline* which process media streams such as audio and video streams; and *non-real-time deadline* tasks, such as QoS management tasks.

2. *Scheduling Algorithm*

   To schedule a set of periodic and deadline-driven tasks we choose a *mixed scheduling algorithm* [11] which combines rate-monotonic and deadline-driven scheduling algorithms. The rate monotony applies to the task processing media/connections according to the sample and packet production and the consumption rate at the devices. The deadline-driven algorithm applies to intermediate tasks such as moving data between devices, where the deadline is less than or equal to the period allocated to tasks responsible for producing/consuming the data.

[5] In the current implementation of the ATM network, there is no admission control in the ATM or AAL layer. Hence, we assume a lightly loaded ATM LAN, which provides an environment where the network resources are available.

[6] The periodic tasks are a subset of deadline-driven tasks, because the task period represents also a deadline.

4.2 OS Resource Availability Tests

The *resource admission service* at each subsystem level tests its own OS resources with a *schedulability test*. The final decision about the end-point OS resource CPU (i.e., if all tasks are schedulable) must be performed by the transport subsystem admission service which has more complete information about resource multiplexing at the end-point.

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

From the schedulability condition for the mixed scheduling algorithm [11] $\sum_i \frac{e_i}{p_i} \leq 1$, where i is a number of tasks, we can derive the schedulability test in the application and transport subsystems:

- *Schedulability Tests in the Application Subsystem*

  Let us assume that $e^A_{o,i,r}$ specifies in application A the task r duration (processing time) of medium i sample (video/robotics data) 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:

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

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

  $$\sum_r e^A_{o,i,r} \leq P_{o,i} \qquad (2)$$

- *Schedulability Tests in 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 switching between network tasks. The schedulability test in the transport subsystem is:

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

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

  The schedulability test - equation (3) - represents the global schedulability test at the end-point for CPU resource sharing.

Both tests assume that there is no interference of other applications and/or users. If an interference time D_I is present, it has to be added to the left side of the equations (1), (2), (3), and (4). The interference can be formally bounded as described in [10], but the current operating systems provide limited means to bound the interference [2] and provide a deterministic behavior. Because of an insufficient support of a determinism in OS, in our implementation we have limited the number of applications and users on the workstation. Further, the context switching contributes to unpredictable behavior [2], therefore the goal is to have a minimal number of context switching. The most predictable case is achieved when tasks are implemented in one process (no process context switching).

4.3 Network Resource Availability Tests

The network resource availability test is needed for end-to-end QoS guarantees. We discuss two tests: an *end-to-end delay test* and a *throughput test.* The final decision of the end-to-end QoS guarantees is performed at the remote (receiver) end-point in the application subsystem admission service, which has the complete information about the application-to-application behavior.

- *End-to-End Delay Test*

  For the *end-to-end delay test*, the parameters of interest are: (1) access and processing delay of a sample at the sender side D^S, which consists of the sum of (a) processing time of all application tasks for the sample and (b) processing time of all network tasks; (2) delivery and processing delay of a sample at the receiver side D^R which is computed similar to D^S, and (3) network propagation and queuing delay D^N. The final end-to-end delay test is performed in the application subsystem by the admission service. The sum of D^S, D^R, and D^N for medium i sample has to satisfy equation:

  $$D_i^S + D_i^R + D_i^N \leq C_i^A \tag{5}$$

- *Throughput Test*

  In the *throughput test* we test that: (1) the throughput over one connection must be less than the total bandwidth of the host interface in that particular direction; and (2) the sum of throughputs over all connection in each direction must be less than or equal to the total bandwidth of the host interface in each direction. For example, our ATM host interface has an effective bandwidth in each direction of 135 Mbits/second. Therefore, the sum of throughputs of all connections for sending data is checked against the 135 Mbits/second bound. The same test is done for connection over which we receive data. These tests are performed at the transport subsystem level. The throughput parameter is then translated into sample rate (sample size) and the schedulability test is done thereby determining if the network throughput can be propagated through the end-point to the application.

5 Information Distribution Services

Distributed networked applications have distributed resources, so QoS parameters as well as local decisions must be propagated between consecutive layers and between corresponding peers.

Layer-to-layer communication includes propagation of responses ('accept', 'reject', 'modify') about the acceptance of QoS between two consecutive layers. Communication between the layers is carried out by the tuning service at the user/application interface and by the QoS translator service at the application/transport interface. Further, if the QoS specification in every layer is different, translation is involved in the layer-to-layer communication as it was described above.

Peer-to-peer layer communication is performed by the *negotiation/renegotiation services.* In peer-to-peer negotiation two separate negotiations are supported:

- *Application QoS Negotiation*

  Application QoS negotiation happens between the application subsystems. It has some general properties, such as exchange of application QoS among the remote sites. It can also include some application-specific properties:

  - in our telerobotics experiment the application QoS negotiation is initiated at the operator side;
  - the operator specifies additionally to application QoS (Figure 2) also a *non-QoS information* (e.g., initial operation 'send me video frame' to evaluate the remote environment) which is sent to the remote robot.

- *Network QoS Negotiation*

  The network QoS negotiation is performed by the transport subsystem and includes: (1) exchange of the network QoS values, and (2) exchange of VCI mappings to specific connections supporting the media transmission.

A detailed description of the QoS negotiation service in a robotics environment is presented in [3].

6 Implementation Issues

The implementation of the service kernel is tested through our driving application – *telerobotics.* The hardware components of the experiment are shown in Figure 10. The end-point communication architecture (Figure 5) is implemented on the IBM RS/6000s. The connections between the robot control and the communication system is achieved through bit3 cards via bus-bus communication. This application puts new constraints on the system architecture of the end-points as well as on communication protocols and services in the network architecture.

As we pointed out earlier, this application has several specific properties which need to be considered in the implementation of the service kernel: (1)

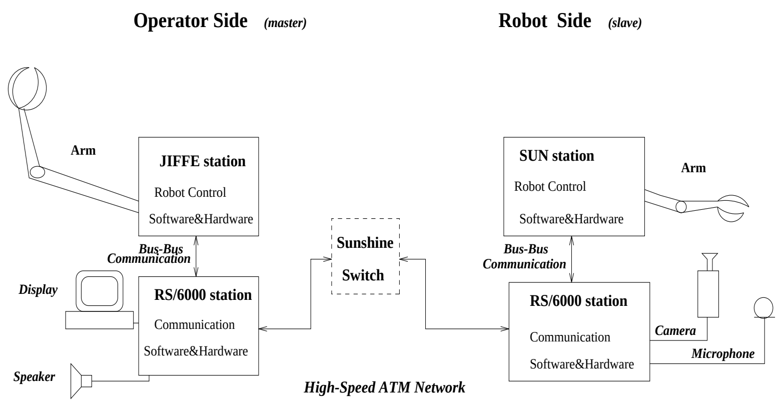

Figure 10: *Telerobotics Experimental Environment*

Telerobotics includes end-points (robots) without a human user, as well as end-points with a human user. Therefore initiation of negotiation, setup, and transmission is asymmetric. It is always started by the operator. (2) The quality requirements for sensory data are very high. (3) Video and audio are supporting information for the operator in order to have audio-visual support control over the workspace of the remote robot, and to make the proper decision in case of a failure or dangerous situation.

The services in the kernel are coordinated by the *QoS Broker* [1], an end-point orchestration entity which schedules the activities of the QoS management.

7 Summary and Conclusion

Translation, admission and negotiation services represent new services in multimedia communication systems and become a necessity for support of the call/connection establishment management if QoS guarantees are required.

The advantages of these services are:

1. translation services allow domains to work in their preferred specification language;
2. admission services in the network are extended to the end-points which implies admission in upper layer protocols, and cooperation between upper and lower layer protocols to make global resource admission decisions; and
3. information distribution services provide communication between the layers and peers during call establishment.

It is important to emphasize that there are also other QoS services which will become necessary for QoS management during the transmission of continuous media. Examples of such services are *renegotiation* , *monitoring*, *notification* , etc. These services will soon become part of the service kernel.

References

[1] K. Nahrstedt, J. Smith, "The QoS Broker", *Technical Report*, MS-CIS-94-13, University of Pennsylvania, March 1994.

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

[3] K. Nahrstedt, J. Smith, "QoS Negotiation in a Robotics Environment", *Proceedings of Workshop on Distributed Multimedia Applications and QoS Verification*, Montreal, Quebec, Canada, May 31-June 2, 1994.

[4] H. Tokuda, T. Nakajima, P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System" *Technical Report*, Carnegie Mellon University, Pittsburgh, PA, 1993.

[5] C. J. Parris, D. Ferrari, "A Dynamic Connection Management Scheme for Guaranteed Performance Services in Packet-Switching Integrated Services Networks", *Technical Report*, UC Berkeley, October 1993.

[6] L.Zhang, S. Deering, D. Estrin, S. Shenker, D.Zappala, "RSVP: A new Resource ReSerVation Protocol", *IEEE Network*, September 1993.

[7] E.W. Biersack, "Performance Evaluation of Forward Error Correction in an ATM Environment", *IEEE JSAC*, May 1993, Vol.11, No.4, pp.631-640.

[8] J. Jung, D. Seret, "Translation of QoS Parameters into ATM Performance Parameters in B-ISDN", *IEEE INFOCOM'93 Proceedings*, Vol.II, San Francisco, CA, March 1993.

[9] A. Mauthe, W. Schulz,R. Steinmetz, "Inside the Heidelberg Multimedia Operating System Support: Real-Time Processing of Continuous Media in OS/2", *Technical Report*, IBM Research Center Heidelberg, Germany, 1992.

[10] K.W. Tindell, A. Burns, A.J. Wellings, "Guaranteeing Hard Real-Time End-to-End Communication Deadlines", *Technical Report Number RTRG/91/107*, Department of Computer Science. University of York, December 1991.

[11] C.L. Liu, James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *Journal of the Association for Computing Machinery*, Vol.20, No.1, January 1973, pp.46-61.