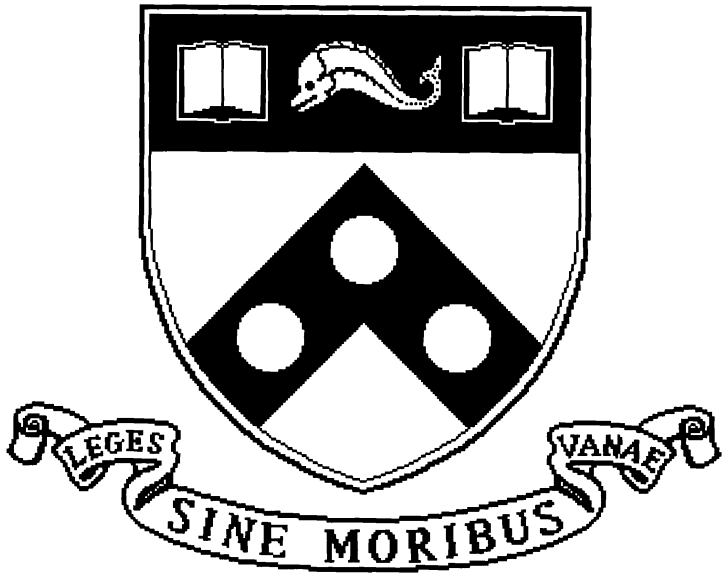
Network Service Customization: End-Point Perspective (Proposal)

MS-CIS-93-100
DISTRIBUTED SYSTEMS LAB 36

Klara Nahrstedt



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

December 1993

Network Service Customization: End-Point Perspective (Proposal)

Klara Nahrstedt

Distributed System Laboratory
Computer Science Department
University of Pennsylvania
e-mail: klara@aurora.cis.upenn.edu

Abstract

An important problem with cell-switched technologies such as Asynchronous Transfer Mode (ATM) is the provision of customized multiplexing behavior to applications. This customization takes the form of setting up processes in the network and end-points to meet application "Quality of Service" (QoS) requirements.

The proposed thesis work examines the necessary components of a software architecture to provide QoS in the end-points of a cell-switched network. An architecture has been developed, and the thesis work will refine it using a "driving" application of the full-feedback teleoperation of a robotics system.

Preliminary experimental results indicate that such teleoperation is possible using general-purpose workstations and a lightly-loaded ATM link. An important result of the experimental portion of the thesis work will be a study of the domain of applicability for various resource management techniques.

Contents

List of Figures

1 Introduction

Conventional networks such as the bus-based Ethernet [38] and packet-switched Internet [39], etc. were designed to carry asynchronous traffic. In new network architectures, such as Asynchronous Transfer Mode (ATM) [11], we observe that asynchronous as well as synchronous traffic can be supported in the model of cell-switching [4]. This possibility provides hope for an integration of different communication paradigms. Further, it provides a good opportunity to support new (complex, long-lived, multimedia) applications which have real-time processing requests and employ media of both bursty and periodic (synchronous) natures. There is considerable diversity in different applications which want to communicate using new cell-switched networks. This is especially true with respect to the services required from the network. We consider mainly ATM networks, putting the following important problem in the foreground of our exploration. That is: *How is it possible to customize the network and the end-points to satisfy the requirements of different applications?*

1.1 Problem Description

We consider the class of *distributed real-time networked multimedia applications* which use different media, one or more with real-time constraints. In our experience, this class of applications is further characterized by long duration of usage, and changing temporal and spatial requirements.

Such distributed applications can certainly be supported using dedicated synchronous lines and real-time systems composed of dedicated processors at the end-points of the network. However, this model is prohibitively expensive, both in resources and economic terms, for most applications [36]. With cell-switched networks, the possibility exists for customizing network and end-point behavior according to application requirements. Thus, a key question is : *how well can the customized system approximate the behavior of dedicated links and controllers.* If a good approximation can be achieved, significantly improved resource usage is possible. If not, then we should identify the obstacles, so that they can be removed.

The set of problems which must be addressed includes:

1. the network (as a whole)
   (a) has varying degrees of asynchronous behavior, e.g., jitter,
   (b) is a shared resource and is thus subject to phenomena such as congestion, and
2. the end-points (e.g., computer workstations)
   (a) are shared among different applications (real-time/non-real-time);

(b) are shared among different media (video/audio/sensory data/network data);

These problems impose resource management difficulties in creating an environment for the class of distributed real-time multimedia applications. Resource behavior should be predictable, in order to assure the application-user about the end-to-end communication system performance [1].

The approach we propose is to:

1. establish *customized*, simplex connections/virtual circuits (VC) in the network,

2. establish *customized*, simplex calls/sessions at the end-points,

where all system/communication components are reserved, so that the synchronous traffic can flow.

In general, *customization* means a provision of an environment which satisfies requirements specified by a customer. Network customization means a provision of an environment where data transmission is performed according to the user/application specification. This environment consists of a set of *services* which implement the data transmission. These services need to be customized which means they need to be configured according to the application requirements. To achieve this goal we use the concept of *Quality of Service* (QoS).

Quality of Service is defined in ISO (International Standard Organization) standards as a concept for specification of how "good" offered networking services are [37]. QoS can be characterized by a number of specific parameters. There are several important issues which need to be considered with respect to the parameterization of the QoS:

- *QoS is specified in individual layers.*

  Because the network architecture is a layered architecture, the services are embedded in different layers and therefore the QoS is specified in individual layers. For example, in Open System Interconnection (OSI) specification, the QoS is provided by the networking layer, and enhanced by the transport layer.

- *Semantics of the QoS parameters specifies the quality of the service.*

  It depends on the set of chosen parameters for the particular service what will be measured as the quality of service. For example, in OSI specification the transport service allows the user to specify parameters such as *connection establishment delay, throughput, end-to-end delay* which means that the quality of the transport service will be measured and evaluated on the

[1] In the class of applications under consideration, the accurate characterization of the performance is the main requirement.

performance of the connection establishment in terms of its delay and on the performance ot the packet transmission in terms of throughput and end-to-end delay.

- *QoS parameters also specify the service objects.*

  Because services are performed on different objects (e.g., connection, virtual circuit (VC)) the QoS parameterization specifies also the quality of the service objects. For example, the transport service performs the transport over connections, therefore the transport connections can be characterized through QoS parameters such as connection establishment delay, throughput of the connection, and end-to-end delay over the connection.

- *Representation of the QoS parameter values specifies the range of the quality.*

  The QoS parameter values can be represented in form of *statistical bounds*, or *deterministic bounds*. The deterministic values can be represented through a *single value* (average value), a *pair of values* (minimal and average value), or an *interval of values* (lower bound is the minimal value and upper bound is the maximal value). Therefore, the representation of the parameter values determines for example if the customized transport connection is able to carry video traffic with variable bit rate (in this case the throughput is specified through deterministic interval of values), or sensory data traffic with constant bit rate (in this case the throughput is specified through a single value).

There are several implications of this customization:

1. a complete specification of parameters needed for establishment of customized session, call, connection, VC [2];

2. a set of call/connection establishment protocols with the notion of dynamic parameter negotiation/renegotiation and feedback paradigm;

3. services for decision-making during the call/connection establishment phase;

4. flexible, dynamic protocol stack for data transmission with enforcement/control mechanisms corresponding to the negotiated parameters;

5. flexible, dynamic management strategies of sessions/calls/connections/ VC's.

[2] In our terms, *session* defines one or more application calls to remote user(s), *call* is defined as point-to-point application connection between sender and receiver and it can map to one or more transport connections. A *transport connection* defines a logical network connection and it can map to one or more VC's. One or more *VC's* map to a physical connection.

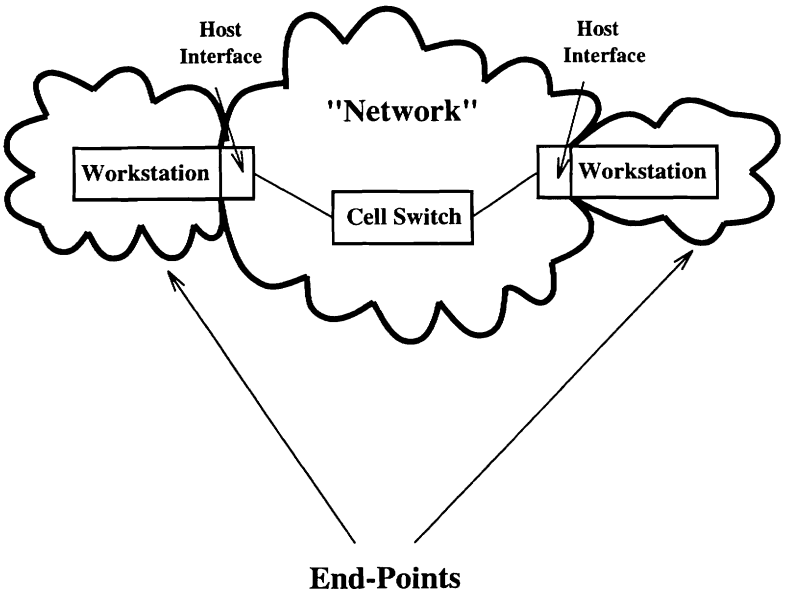


Figure 1: *End-Points Versus Network*

1.2 Assumptions and Restrictions in our Thesis Work on the Problem

We restrict the problem space with a set of assumptions about the role and capabilities of other system elements. We also set our proposed focus and emphasis.

1. **The network includes mechanisms/primitives for connection customization.**

   We will assume that the network has the proper mechanism for sharing the resources, and that resource reservation at the level of VC's is provided in the ATM Adaptation Layer (AAL) [11] and below. The network customization is done through *service class* provision in the AAL layer.

2. **The main objective is the customization at the end-points.**

   We are interested in customization issues at the end-points of a networked multimedia system (Figure 1). The performance of the end-points is equally important in comparison with network performance otherwise the networked multimedia system can not provide a systemwide guarantee to the user or application. It is important to properly design end-points which can support different types of traffic and provide local guarantees towards the global solution of customization.

3. **The communication hierarchy at the end-point consists of two major subsystems.**

   We will assume that the end-points include two major subdivisions of the communication hierarchy - an *application subsystem*, which represents the application layer, and a *trans-*

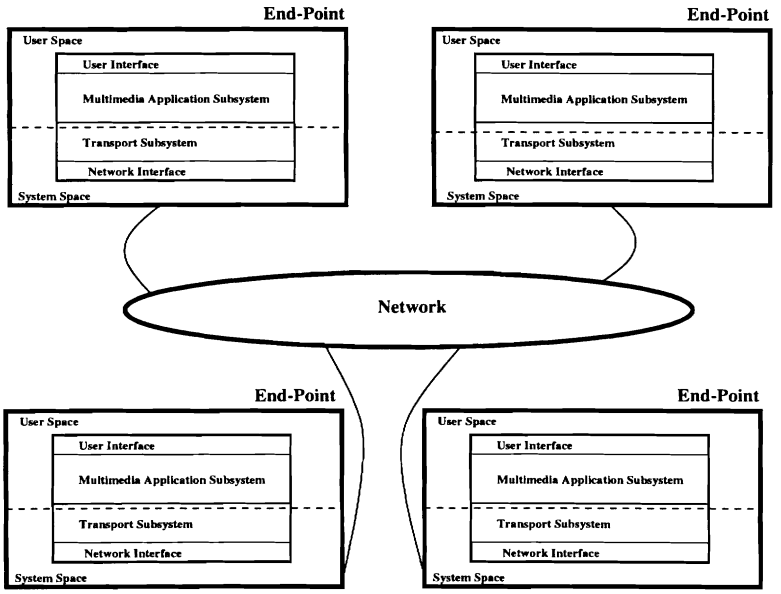


Figure 2: *Application/Network Model at the End-Point*

port subsystem, which represents the transport and network layers. The application/network model at the end-points is shown in Figure 2.

4. **An application-driven approach is taken toward customization.**

   The focus is on application-driven approach in order to provide customization which means that the application requirements determine mainly the configuration of the services and their QoS parameters at the end-points. This approach implies (1) specification of application requirements, (2) parameter specification of application and network services involved in the provision of customized calls and connections, (3) parameter specification of local system resources at the end-points, and (3) communication mechanisms for *balance* among the application requirements, the local system resources at the end-points, and the network resources.

5. **QoS parameter management is analyzed in order to provide customization.**

   Taking the QoS parameterization approach, a new framework (parameter specification, services, protocols) is necessary for provision of Quality of Service in the end-point architecture. The provision of QoS in the end-point architecture includes three parts:

   - **QoS parameter management** manages all parameters, which specify application, network and local operating system characteristics for balancing the requirements/guarantees ratio. These parameters at the end-points are stored in a *QoS parameter database*. The

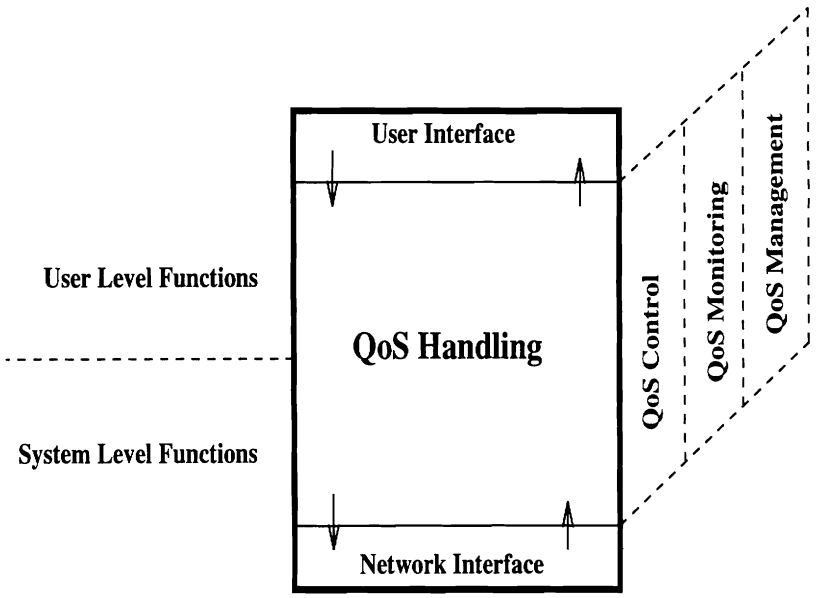


Figure 3: *QoS Handling in the End-Point*

management provides and uses service primitives and protocols for QoS parameter set up, QoS parameter distribution, QoS parameter change, QoS parameter retrieval, and QoS parameter admission.

- **QoS parameter monitoring** is part of the communication management process. Here the actual QoS parameter values, which are agreed on, are monitored during the transmission phase. This functionality can be used for performance evaluation and based on the evaluation a change of QoS parameters can be enforced.
- **QoS control** (e.g., error control, ordering control, rate control) is part of the transmission process where different mechanisms for enforcement of QoS are embedded in transmission protocols.

We call the three parts for provision of QoS the *QoS handling* and they are shown in Figure 3.

We restrict our attention to QoS parameter management, its functionality, new services, primitives and protocols, as well as architectural issues, i.e., placement of QoS management in the end-point architecture.

1.3 Why Hasn't the Problem Been Solved?

There are two main reasons why the problem hasn't been solved: (1) the network research has been split from the application research, and (2) the cell-switching/ATM is a new technology. We

briefly discuss these two reasons from the research and experimental viewpoints. Deeper analysis and review of the current research is given in section 4.

- **The network research has been split from the application research.**

  The reason for the split can be seen in the different objectives of the networks and the applications.

  The network research emphasis is on achieving higher speeds and developing, towards this goal, new technology with hardware and low level software solutions. The end-points are considered traffic senders and receivers specified by their traffic and performance characteristics.

  The application research emphasis is on provision of sophisticated methods for solving different user-oriented problems. The applications use for their solutions different media, as well as different communication, control, and decision-making mechanisms. The network is considered a transport medium with certain (static) properties.

  Further, the QoS research is typically done from the network viewpoint because at this level the services are completely specified. The transport, network, and currently AAL services are specified through the parameters; for example, throughput, delay, and loss rate. For these parameters there have been developed different algorithms, protocols, and services for control and enforcement of the parameter values.

  The application services are not parameterized in the same way as the network services are, probably due to their variety, and there is no bidirectional relation and communication between them. This means that the applications at the end-points have only a few possibilities or none to specify through the API (Application Programming Interface) their requirements to the network and get feedback on what the network can offer.

- **The cell-switching/ATM is a new technology for provision of high-speed Broadband Integrated Service Data Networks (B-ISDN).**

  The cell-switching/ATM technology is a new technology which is not yet widely available. Current available network access to every user is LAN Ethernet or WAN Internet. Operational ATM networks in the form of testbeds are available to only a few research laboratories (e.g., our laboratory is part of the AURORA testbed and therefore has access to an ATM network.).

  The main emphasis in developing this new technology has so far been (1) hardware (e.g., switches, host interfaces) because it is crucial for the new high-speed networks, and (2) low

level software (e.g., host interface device drivers) in order to provide access to the hardware or to new adaptive control and communication mechanisms in the lower layers of the B-ISDN.

1.4 Outline of the Proposal

Section 2 presents the functionality of QoS parameter management as an additional component in the end-point architecture. Section 2.1 elaborates on the QoS parameterization in the end-point, its split into three domains, and its representation in a QoS parameter database. High-level structures for QoS parameters from each domain are briefly presented. Section 2.2 describes basic communication mechanisms to fill the QoS parameter database. Section 2.3 discusses task scheduling at the end-points as one of the major criteria for providing QoS guarantees to the application.

In section 3 we present an end-point architecture, services and protocols which support QoS parameter management. We describe in section 4 related work in the QoS area.

Section 5 provides an overview of an experiment with which we want to test the QoS approach and the soundness of the proposed architecture. Section 6 outlines the expected results. Section 7 gives a plan of steps and measurements in the thesis work.

2 QoS Parameter Management

QoS parameter management is a necessary new functionality in communication systems for customization support. The reason for the introduction of this component is the diversity of networked multimedia applications using high-speed ATM networks. With incoming new media and applications it is not effective or flexible to develop special-purpose protocols or to use special-purpose machines for every application. The environment we are facing is one in which users want to run distributed multimedia applications on general purpose machines connected through high-speed ATM networks.

Specification and implementation of QoS parameter management involves three tasks:

- **the QoS parameters at the end-points have to be specified**;

  In order to manage QoS parameters we need to parameterize the end-point system and specify the QoS parameters. The end-point system consists actually of three components as it is shown in Figure 2. The application and network components are represented through application and transport subsystem. These subsystems are embedded in the environment

of the local operating system (third component). This division justifies the separation of the parameters into:

- *application QoS parameters*, which represent the application requirements, application services and their resources,
- *network QoS parameters*, which parameterize the network services and their resources,
- *system parameters*, which parameterize the local system services and their resources.

All parameters are stored in QoS parameter database and we will provide primitives which *set up/change* and *access* the database. We will describe the representation of all parameters in section 2.1.

- **the QoS parameters have to be exchanged among different system components**;

  The parameter split implies new paradigm in exchanging the parameters such as negotiation, renegotiation, and translation among application, network, and local operating system. We will address these issues in section 2.2.

- **the third task is to find tests which support predictability in order to guarantee QoS parameter values.**

  This task requires global parameter view of the end-point system in order to guarantee QoS parameter values. The global parameter view leads us to the introduction of a unifying system model, where the application, network and local system parameters meet and together describe the global environment with respect to the running application, network, and local system resources. Further, this task requires a detailed analysis of the operating system support with respect to the scheduling capabilities and other resource management mechanisms. We will discuss these issues in section 2.3.

2.1 Parameter Specification

We understand QoS in a broader sense when we look at it from an application ("user") perspective. In the earlier stage QoS was only understood as the quality of network service. We extend this notion also to quality of application services as well as quality of network service because more and more multimedia applications are included in a distributed environment and they become an integral part of the networking environment. Also with a variety of applications, their media and their requirements, it is necessary to consider quality-of-application services in order to provide a more dynamic solution to the customization of communication end-points.

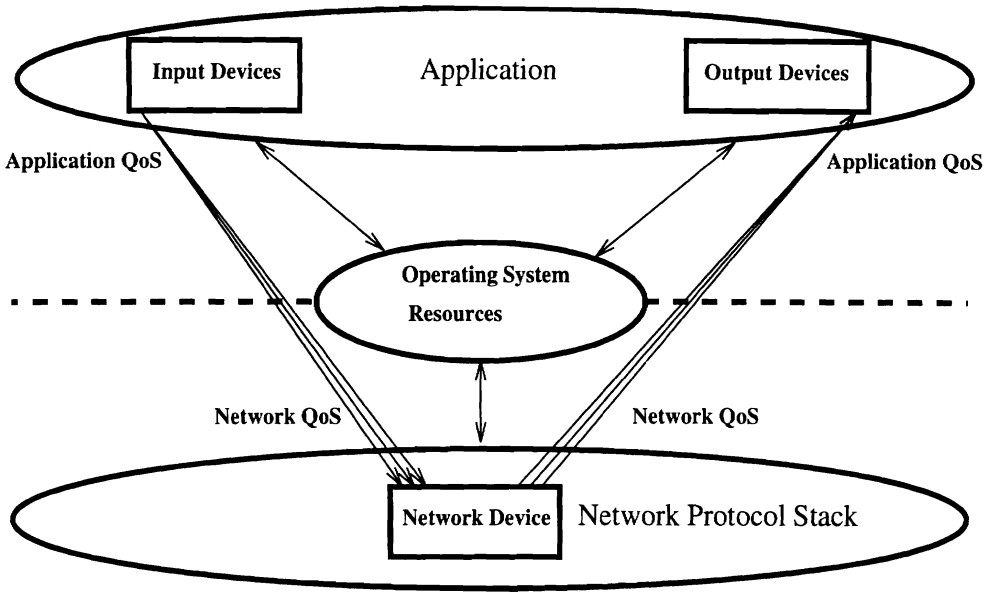


Figure 4: *QoS Model*

Further, the application and transport subsystems are embedded in an operating system environment, therefore the application QoS and network QoS depend on the system services and their quality.

The introduction of application QoS parameterization and local system resources parameterization requires careful analysis of QoS in general.

The split of QoS parameters is represented through the QoS model shown in Figure 4.

2.1.1 Application QoS Parameterization

Application QoS is "quality" in terms meaningful to application services. Application services (e.g., synchronization, presentation) perform activities over *media*, therefore media are the service objects and we will characterize them. Application-characteristic parameters include information on multimedia *media quality* and *media relations* (Figure 5). *Media quality* describes the parameters of the input and output devices.

More precisely, for each media device the user specifies:

- media characteristics: *device identification, sample size, sample rate, compression method and its parameters*;

  Parameters such as sample size and sample rate are numerical values and will be represented as deterministic values.

- transmission characteristics: *end-to-end delay of the sample, sample loss rate, cost*;

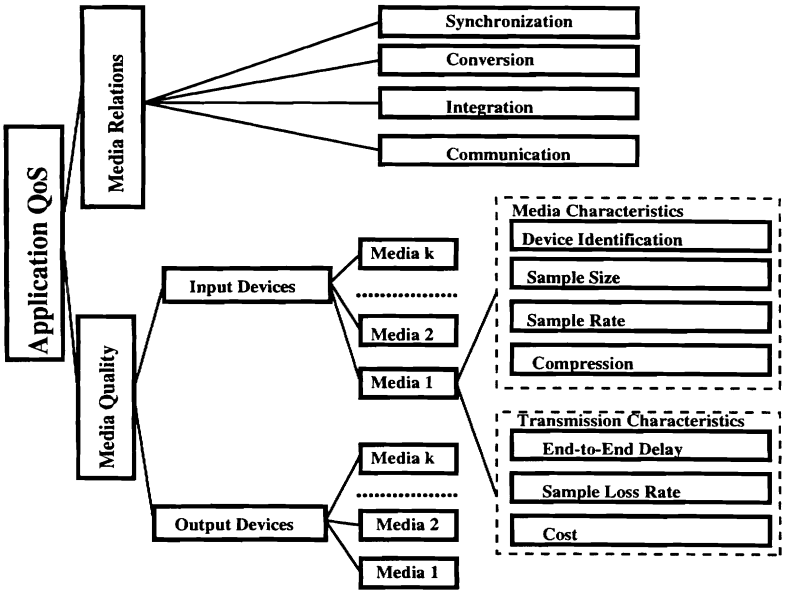


Figure 5: *Application QoS Parameterization*

These parameters will be represented through their deterministic bounds.

Media relations describe specification of intermediate tasks which provide relations among media as media conversion, synchronization, communication (unicast, multicast), integration (interleaving/multichannel variant of transmission), etc.

With respect to the communication service at the application layer *application QoS parameters* represent global description of the logical connection (service object) *call* between application sender and application receiver.

The configuration of the application QoS parameters comes during the initialization phase either from the user using interactive user interface or from the application installation program where the application specification is stored. During the call establishment phase the configuration of the application QoS parameters comes from (1) the remote application QoS parameters where the possible parameter values of the output media quality are encoded, (2)the network QoS parameters where the information about the possible parameter values of the input media quality is encoded.

2.1.2 Network QoS Parameterization

Network QoS is "quality" meaningful to network and transport services. The QoS parameters specify the network connection characterized through *throughput pledge*, message stream with its *traffic characteristics* which goes through the network connection, and its *performance requirements* (Figure 6).

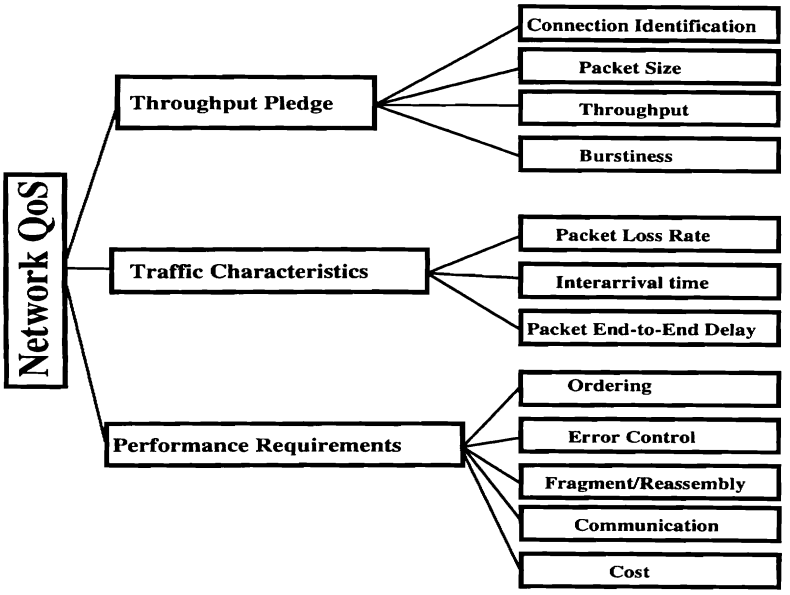


Figure 6: *Network QoS Parameterization per Connection*

More precisely, the network QoS specification includes:

- throughput pledge: *connection identification, packet size, packets/second (throughput)* and *burstiness.*

  The parameters such as packet size, throughput and burstiness are given in terms of their deterministic bounds.

- traffic characteristics: *delay bounds, interarrival time, loss rate*;

  These parameters are specified through their deterministic bounds.

- performance requirements: *ordering, error control, fragmentation/segmentation, cost computation, communication (unicast/multicast/broadcast).*

  These parameters specify the protocol tasks which have to be performed in order to guarantee network QoS, and they are specified either through BOOLEAN values (e.g., ordering is required or is not required) or through specification of the mechanism (e.g., error control requires retransmission/forward error correction/no error correction).

The configuration of the network QoS parameters comes (1) from an installation program where the information about the possible parameter values (maximal bounds) is encoded, and (2) from the application QoS parameters where the information about the requested parameter values is encoded.

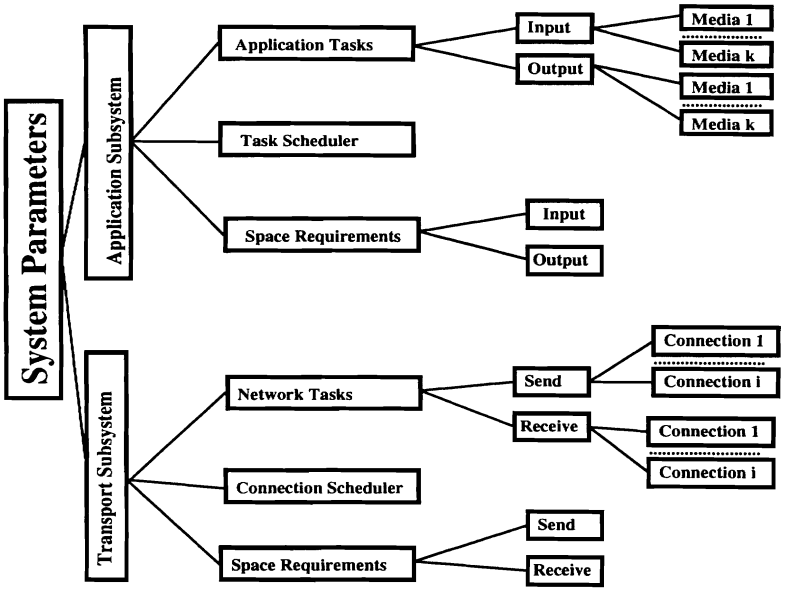


Figure 7: *End-Point System Parameterization*

2.1.3 Local System Parameterization

System parameters represent the end-point operating system parameterization (e.g., buffers, tasks implemented as threads or system process). They specify the system resources needed by the :

- **application subsystem** with *tasks* which have to be performed per media and their processing time, *task scheduler* with respect to input/output processing of different media, as well as *space requirements* for each media taking into account the direction (input/output).
- **transport subsystem** with *tasks* performed on each connection and their processing time, as well as the *connection scheduler* and *space requirements* allocated to each connection during the transmission.

Figure 7 shows the global structure of the end-point system parameterization.

The configuration of the local system parameters comes from (1) an installation program where the configurable application/transport protocol tasks and their processing times are filled in, (2) the application QoS parameters where the application space requirements and task selection are filled in during the call set up phase, and (3) the network QoS parameters where the network space requirements and task selection are filled in during the connection set up phase.

Computation of the task scheduler needs information such as the selected protocols tasks, their processing time, and the chosen scheduling algorithm which determines the selection of the task priorities.

A more detailed parameterization of tasks, space requirements, and the schedulers is described in subsection 2.3.1.

2.2 Parameter Communication

Under QoS parameter communication as part of QoS parameter management, we understand the functionality to set up the QoS parameter database at every end-point participating in the distributed networked multimedia application.

The split of QoS parameters at the end-points shows that the application QoS parameters and network QoS parameters are clearly different ways of talking about the behavior of a communication system. We must provide mechanisms for communicating parameters in the appropriate language among the application end-point entities as well as between the application and network entities and between application/network entities. Further, these parameters have to be mapped onto local system resources.

A general architecture for communication of QoS parameters requires two services : **negotiation/renegotiation of QoS parameters** and **translation of QoS parameters**. We will describe briefly the functionality of both services.

2.2.1 Negotiation/Renegotiation of QoS Parameters

To characterize an actual negotiation, we ask *who are the parties ?*, and *how do the parties negotiate?*. There are really two parties to any QoS negotiation. We will consider *peer-to-peer negotiation*, which can be, for example, application-to-application negotiation or network-layer-to-network-layer negotiation, and *layer-to-layer negotiation*, which might be, for example, application-to-network negotiation, or human-user-to-application negotiation. Figure 8 shows a communication paradigm of how peer-to-peer negotiations are glued together through layer-to-layer communication. Layer-to-layer communication is represented by a *QoS broker* which implements the application-to-network negotiation. A more detailed discussion of negotiation protocols can be found in section 3.2.

Here, we will present a brief discussion of the functionality which the negotiation process should include:

1. **peer-to-peer negotiation**

   Communication of application QoS parameters, respectively of network QoS parameters among remote entities in the same layer is done through a *negotiation protocol*, which can be part of the call/connection establishment protocol (piggybacking the necessary information

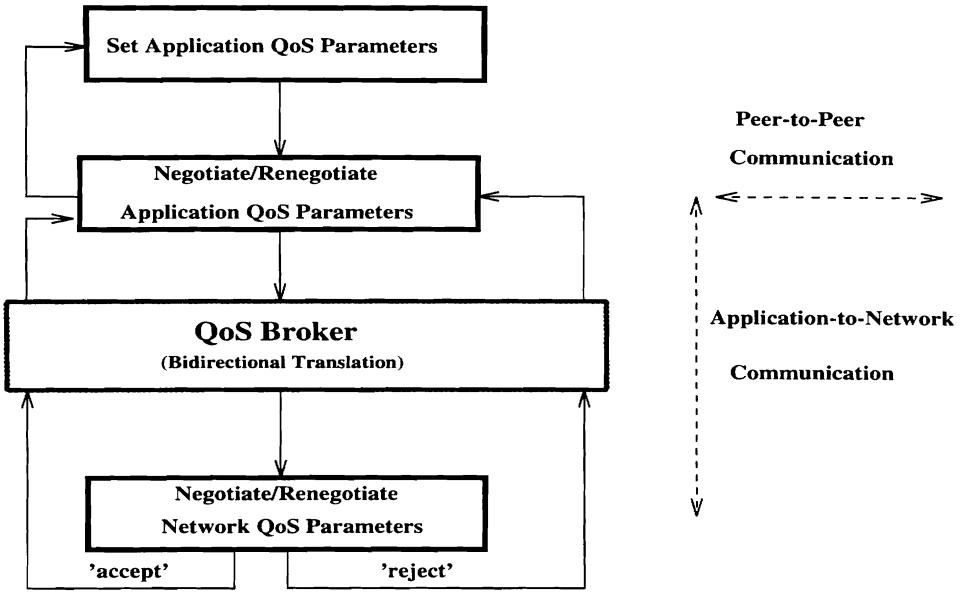


Figure 8: *QoS Parameter Communication*

in the call set up message); or after a call is set up, the negotiation of QoS parameters can start separately. This process establishes an agreement between the parties with respect to the peer QoS parameters. In our approach the QoS parameters are carried in the message of the call set up PDU (Protocol Data Unit).

Further, in our approach the application negotiation is independent of network negotiation, i.e., the negotiation of application QoS uses a non-real-time, non-parameterized transport connection for communication of application QoS parameters between the application end-point entities. If this negotiation is successful, the network QoS negotiation follows in order to set up a parameterized transport connection where the actual data will flow through. The advantage of this approach is that the decision about acceptance is done separately, and if the application end-points can not accommodate the requirements, the network negotiation can be avoided. The disadvantage is that if we go through the negotiation between application entities, and an agreement is reached, but the network can not accommodate the application requirements, then the first negotiation step is a waste of time and resources.

It is possible also that application and network negotiations can be done in one step. That is if the application QoS parameters are translated into network QoS parameters and customized network connections are negotiated, then the result of the network QoS negotiation for each connection is reported to the receiver and there translation occurs from the set of accepted network QoS parameters into the possible application QoS parameters. Now the

acceptance check for the application QoS parameters follows at the receiver. The advantage of this approach is that we get an acceptance answer in one step and don't need an additional non-parameterized connection for the negotiation of application QoS parameters. The disadvantage is that if we reserve network resources and at the end the receiving application does not have the resources, we waste network resources. The trade off between both approaches will be the cost of wasting resources (application versus network) in the case of negative acceptance. We argue that wasting network resources is more expensive than wasting end-point application resources because the network is shared among a much larger number of users/applications than end-points. Therefore our approach to separate the negotiation into two steps is a 'cost saving approach'.

2. **layer-to-layer negotiation**

   - **application-to-network negotiation**

     This negotiation includes the exchange of QoS parameters between the application and the network. Negotiation can be understood also in more general sense as the exchange of parameters between two consecutive layers in the network hierarchy (in this case the application is part of the network hierarchy). Further, it includes the agreement between what an application requires and what the network can provide.

   - **human user-to-application negotiation**

     The negotiation process between a human user and an application has an interactive character. It is an optional process because in case where the end-point users are robots, the application requirements are encoded in the machine and no interaction is possible. But for a human interface this negotiation is of importance in determining the quality of audio/visual information. The negotiation can use audio/video clips or a simulation to set QoS parameters at the human user level.

Renegotiation is a process of negotiation when a call is already set up. This activity can be initiated if the QoS guarantees either in the application/user or in the network changed over time. The possible signaling for renegotiation is shown in Figure 9.

2.2.2 Translation of QoS Parameters

The split of parameters into three domains according to our QoS model (Figure 4) requires a translation function among these parameter domains. The required mappings and services where

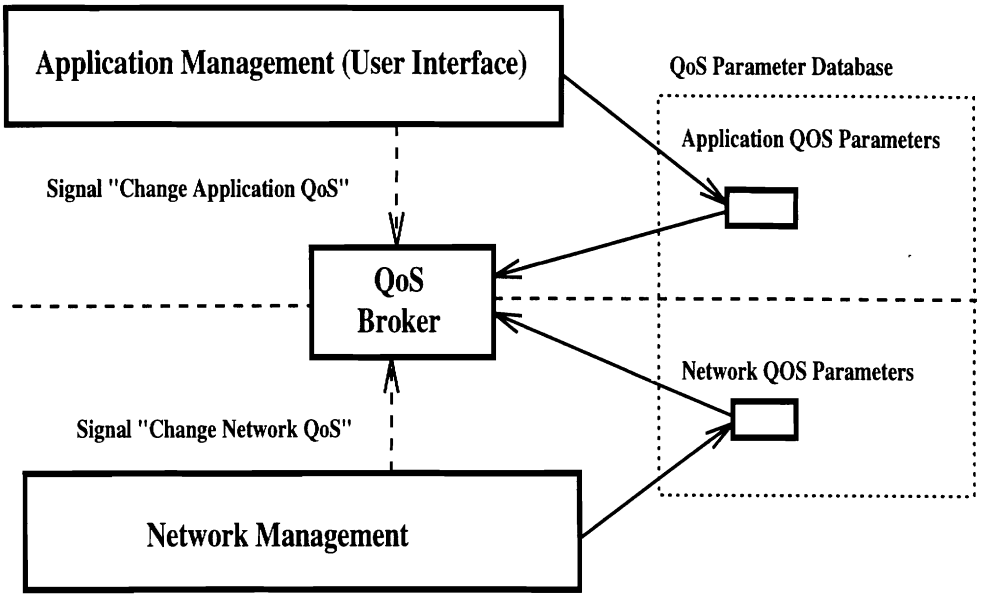


Figure 9: *Signaling Paradigm for Renegotiation*

the mappings are embedded are shown in Figure 10. We will discuss possible translations in section 3.1.

2.3 Guarantees

QoS guarantees have to be met in the application as well as in the network in order to get the acceptance of the users of networked multimedia systems. There are at least three constraints which have to be satisfied in order to provide guarantees:

- bandwidth constraints;
- time constraints;
- space (e.g., system buffer) constraints;

These constraints can be satisfied at the end-points if proper *resource and task management* based on QoS parameters is available.

To work out the QoS guarantees in the application subsystem as well as in the transport subsystem for our class of applications, we need a global view on the end-point system. We will introduce a *system model* which unifies the three end-point domains (application subsystem, network subsystem, and local operating system). This model gives us the parameterization of the end-point which we need for our analysis of QoS guarantees. The parameter values in the system model come from the application, network, and local system parameters (Figure 5,6,7) presented in

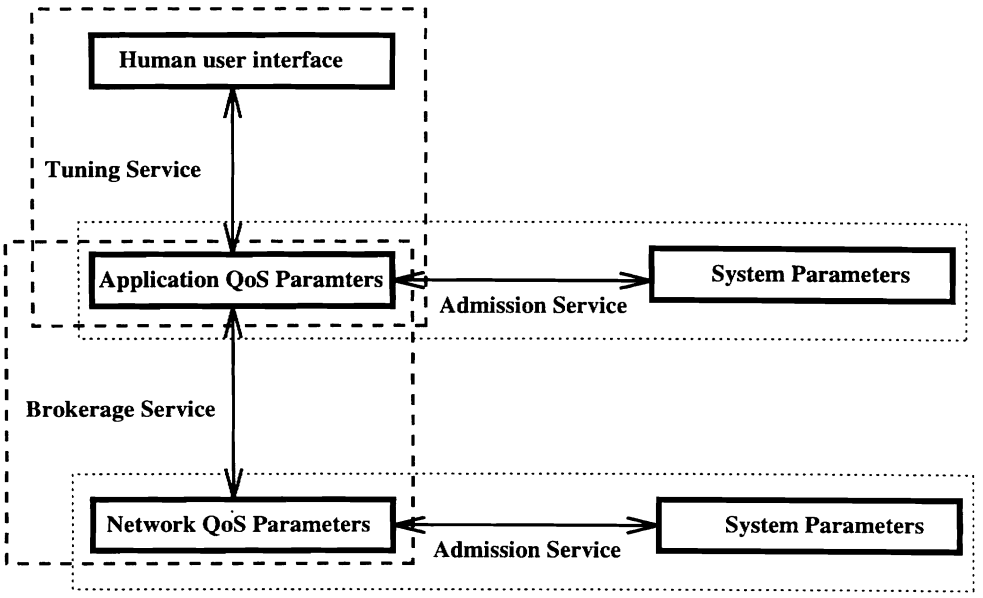


Figure 10: *Mappings among the Parameters*

section 2.1 using the communication mechanisms (Figure 8,10) presented in section 2.2. Through the system model we will show that the application, network QoS parameters, and the local system parameters together provide a (complete) parameterization of the end-point system, therefore we can derive tests which will provide guaranteed QoS parameter values .

2.3.1 End-Point System Model

The system model of the end-point (Figure 11) is based on the scheduling model of real-time tasks described in [29].

Our end-point system model consists of four elements:

- **resources**

  A resource is an entity with a finite capacity. Our resource model is based on the model of Linear Bounded Arrival Processes (LBAP)[29]. In this model a distributed system is decomposed into a chain of resources traversed by the messages on their end-to-end trips. A LBAP is a message arrival process at a resource defined by three fixed parameters:

  - M = Maximum message size (bytes/message)
  - R = Maximum message rate (messages/second)
  - B = Maximum Burstiness (messages)

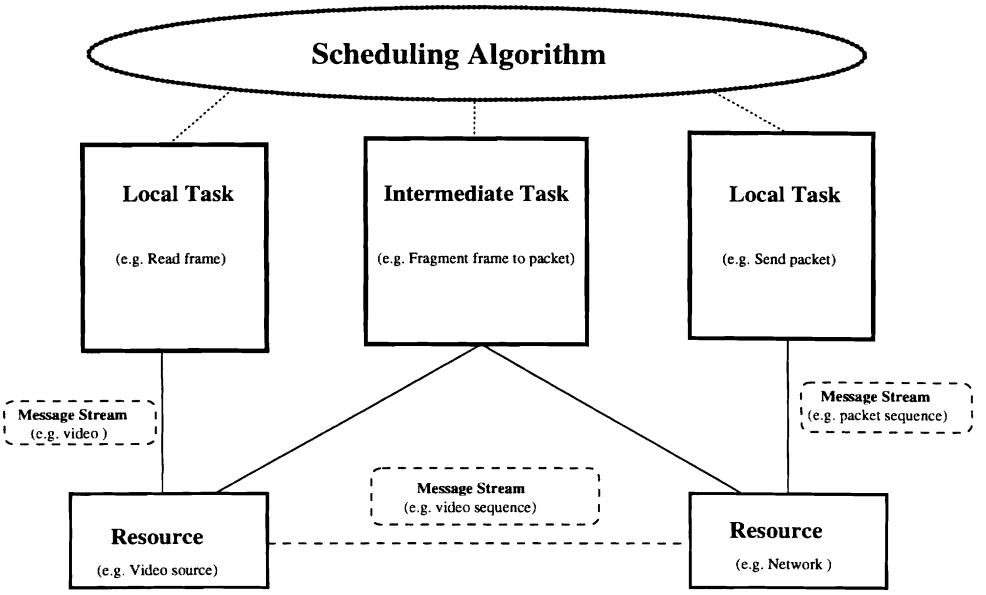


Figure 11: *System Model*

Resources in the application QoS specification are the input and output devices with their media quality characteristics. Resource in the network QoS specification is the network device with its connection throughput pledge. In the local system specification the space requirements represent space resources (buffers).

- **tasks**

  A task is a schedulable entity of the system. It can be invoked to perform a particular function on a resource. In a real-time system it is characterized by its timing constraints. We are considering *periodic* tasks and *deadline-driven* tasks. The timing constraints of the tasks T are characterized by the following set of parameters:

  - *s : Starting point of T;*

    Starting point is the first time a task has to be performed.

  - *e : Execution time of T;*

    Execution time is the time for processing a task on a resource. We assume that the objects the task is processing are *messages* in a message stream.

  - *d : Deadline of T;*

    Deadline of is the time a task has to be completed.

  - *p : Period of T;*

    Period is the maximal processing time of each message.

Further, we partition the tasks at the end-points into three basic classes using deadline times as the partitioning relation:

- hard real-time deadline tasks which process media streams such as tactile and kinesthetic data;
- soft real-time deadline tasks which process media streams such as audio and video streams;
- non-real-time deadline tasks;

We define a task to be *real-time*, if it must be completed by a deadline. We call a real-time task *soft real-time*, if in some cases missed deadline may be tolerated as long as (1) not too many deadlines are missed, and/or (2) they are not missed by much. In the case when a real-time task can't miss its deadline, or else the system fails, we call it *hard real-time.*

There are several reasons why time is the partitioning factor; the fact that sensory data is continuous and the observation that both operating system and resource allocation use time-division multiplexing as a mechanism to implement their resource-sharing paradigms.

Partial information about the protocol tasks are encoded in the application QoS parameters (the media relation part in Figure 5). Partial information about the network protocol tasks are encoded in the network QoS parameters (the performance requirements in Figure 6). More local information about the tasks connected with the application subsystem and the transport subsystem are encoded in the local system parameters (application tasks, network tasks in Figure 7).

- **message streams**

A message stream is an object that resources produce or consume and that tasks process. We deal with a periodic message stream [3]. This stream can be characterized as S = (C,P,D), where

- C is the maximum amount of time required to transmit a message in the stream (end-to-end delay of a message).
- P is the interarrival period between messages in the periodic (synchronous) message stream.

[3]The continuous media are approximated by discrete sequence of messages which are produced/consumed in constant time intervals.

- D is the relative deadline of messages in the stream. It is the maximum amount of time that may elapse between a message arrival at a resource and its transmission (guaranteed logical delay of a message).

The information about the message streams is encoded in the application QoS parameters (transmission characteristics in Figure 5) about the application message stream such as a video or robotics message stream, and in the network QoS parameters (traffic characteristics in Figure 6) about the packet message streams.

- **scheduling algorithm**

The function of a *scheduling algorithm* is to schedule a given set of tasks. The goal is to schedule the set of tasks such that the temporal and spatial constraints are satisfied. This means we need to compute a *feasible schedule* for executing the tasks. If a feasible schedule exists then a scheduling algorithm, applied to a given task set, satisfies a property called *schedulability.* It means if the feasible schedule ensures that each task finishes processing prior to its deadline then a given task set is *schedulable.*

Therefore, to guarantee a task deadline, we need to compute a schedule and check if this schedule is feasible and therefore check the schedulability property.

In order to check the schedulability property over a given set of tasks, we need a performance metric. One of the performance metrics for a real-time scheduling algorithm is the processor utilization. This is the amount of processing time used by a guaranteed task (execution time of a task i - e_i) versus the total amount of processing time used by a guaranteed task (period of a task i - p_i):

$U = \sum_{i=1}^{n} \frac{e_i}{p_i}$

where 'n' is the maximal number of tasks in a given task set.

We consider in our end-point system (Figure 11) periodic and deadline-driven tasks. For scheduling periodic and deadline-driven tasks, we choose *mixed scheduling algorithm* [40] which combines a rate-monotonic and a deadline-driven scheduling algorithm. For both algorithms schedulability tests were developed in [40].

For a rate-monotone algorithm the test is

$U \leq n \times (2^{\frac{1}{n}} - 1)$

For a deadline-driven algorithm the test is

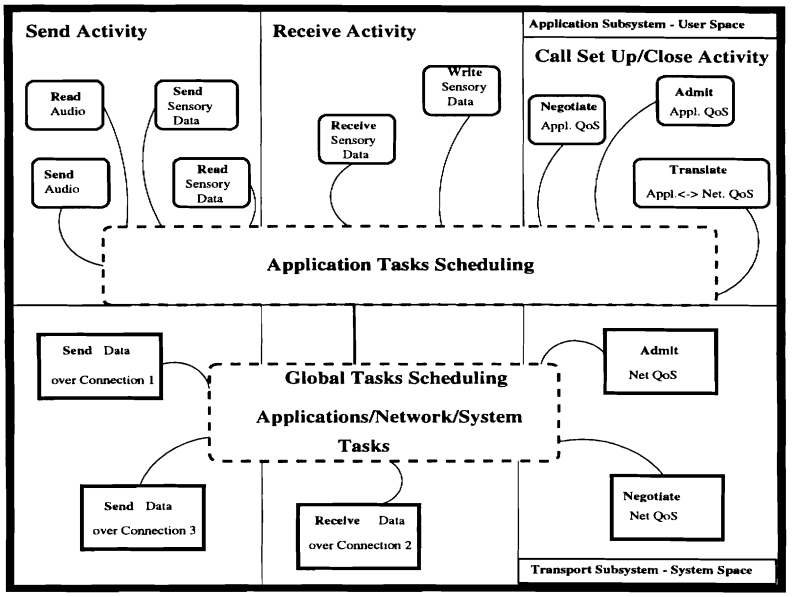


Figure 12: *Scheduling at the End-Point*

$U \leq 1$

Further, we will refine the schedulability test for the application subsystem and the transport subsystem at the end-points.

A computation of feasible schedulers on the basis of the chosen scheduling algorithms for scheduling application subsystem tasks and transport subsystem tasks has to be done. The result is stored in the local system specification (Figure 7).

2.3.2 Scheduling

If we apply the system model to the modeling of the class of real-time multimedia networked applications at the end-points, we have to deal with tasks, resources, message streams and scheduling in the application subsystem, which is embedded in the user space, and with tasks, resources, message streams and scheduling in the transport subsystem, which is embedded in the kernel (system) space of the end-point.

Because of the layered architecture (Figure 2) at the end-point, the scheduling and the computation of a feasible schedule have to be divided, although there are important links between the layers as we will describe further. Figure 12 shows the activity and task scheduling at the end-point. The layering implies that the *schedulability test* has to be performed in the application subsystem as well as in the transport subsystem.

The application subsystem has to schedule its own application tasks, and locally the application

subsystem can decide if its application tasks are schedulable. We will call the schedulability test in the application subsystem the *local schedulability test.*

When it comes to the schedulability decision of the transport subsystem, the schedulability test has to take into account the processing time of the application tasks as well as the transport tasks in order to give end-point QoS guarantees. (The OS scheduler, which schedules all end-point tasks, is shared among all tasks (application and transport subsystem tasks).). Therefore the schedulability test in the transport subsystem decides the schedulability of all tasks running at the end-point and we will call the schedulability test in the transport subsystem *global schedulability test.*

2.3.3 Schedulability Tests

We will refine the schedulability tests described above.

- **Local Schedulability Test**

  The local schedulability test will be performed in the application subsystem by the *application admission service.* Using the local schedulability test, the admission service checks the temporal resources of the application which are specified through the application QoS parameters (Figure 5) and mapped into the system resources (Figure 10). If the local schedulability test is successful, the admission service *reserves* the application temporal resources (i.e. stores the feasible schedule in the QoS parameter database (system.application_subsystem.task_scheduler - Figure 7)). If the local schedulability test is not successful, the admission service rejects the admission request and the application QoS parameters have to be relaxed (degradation of parameters).

  We will assume that the execution time of each application task is $e^{App}_{o,i,r}$, where 'App' indicates that we are dealing with application tasks, 'r' index enumerates the tasks (e.g., receive media, synchronize media) performed over media 'i' (e.g., video, audio, robotics data) in direction 'o' (e.g., input/output). The periodic local tasks such as *read media*, *write media* have periods $p_{o,i}$. The deadline-driven intermediate tasks such as *fragment message* have deadlines $d_{o,i}$, where $d_{o,i} \leq p_{o,i}$. Depending on the implementation of the application subsystem tasks we might have context-switches time cs^{App}_j, when some application tasks are grouped to OS processes. The schedulability test for the tasks which have to be scheduled during the time interval $min_{o=(in,out),i=(1,...N)}(p_{o,i})$ has the following form:

  $$TimeOfAppTasks = \sum_{o=(in,out)} \sum_{i=1}^{N} \sum_{r=1}^{S} e^{App}_{o,i,r} + \sum_{j=0}^{K} cs^{App}_j \leq min_{o=(in,out),i=(1,...N)}(p_{o,i})$$

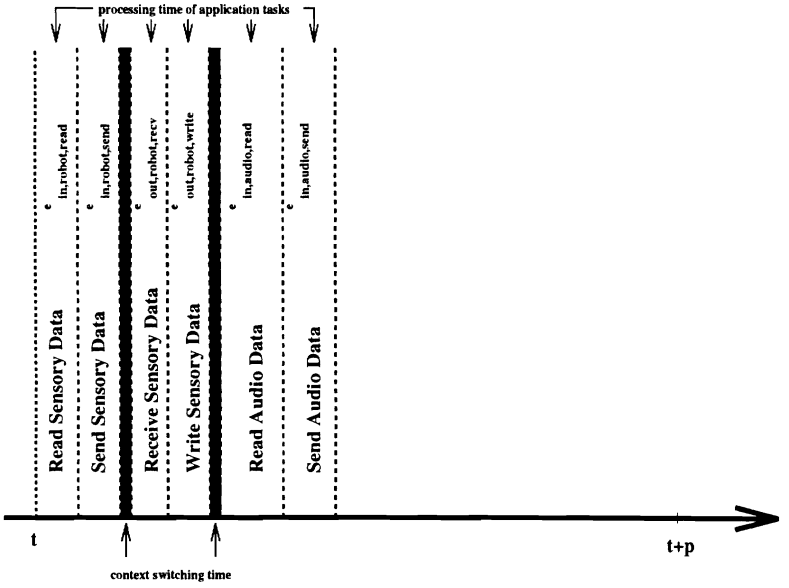


Figure 13: *Scheduling of Application Tasks (Example)*

where the (in, out) index represents the flow direction, N is the maximal number of media supported in the application, S is the maximal number of tasks performed over media 'i' in direction 'o', and K is the maximal number of context switching during the minimal period. Example application tasks *read sensory data, read audio data, write sensory data, send sensory data, send audio data, receive sensory data* from Figure 12 are scheduled according to the mixed scheduling algorithm as is shown in Figure 13.

- **Global Scheduling Test**

The global scheduling test will be performed in the transport subsystem by the *network admission service.* Using the global scheduling test, the admission service checks the temporal resources of the transport subsystem with consideration of delays caused by the application subsystem. The temporal resources are specified through the network QoS parameters (Figure 6) and mapped into the system resources (Figure 10) of the transport subsystem, and through the system resources reserved to the application subsystem (Figure 7). If the global schedulability test is successful, the admission service the admission service *reserves* the temporal resources (i.e. stores the feasible schedule in QoS parameter database (system.transport_subsystem.task_scheduler - Figure 7)). If the global schedulability test is not successful, the admission service rejects the admission request, and the network QoS parameters have to be degraded.

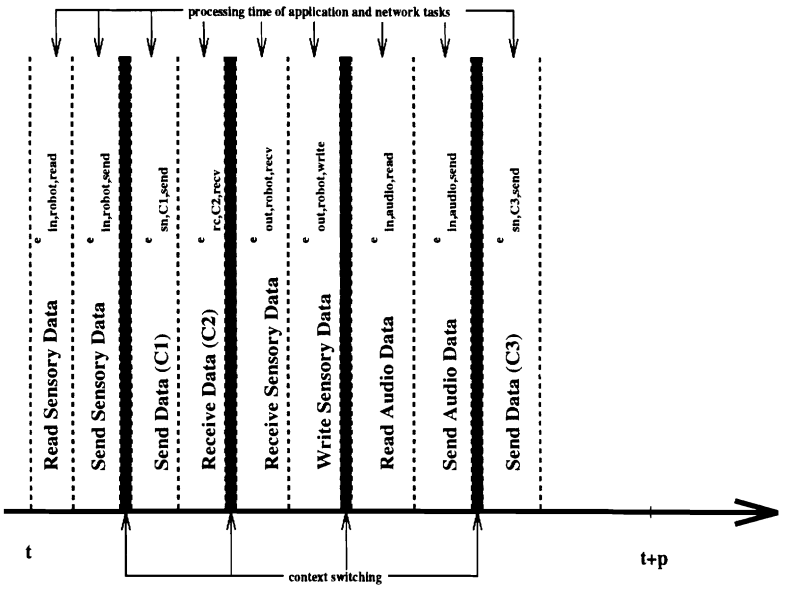


Figure 14: *Scheduling of End-Point Tasks (Example)*

We assume that the execution time of each network task at the end-point is $e^{Net}_{o,j,r}$ where 'Net' specifies the network tasks which performe over connection 'j' in direction 'o'. Depending on the implementation of transport tasks in the transport subsystem we might have the context-switching time cs^{Net}_k. The scheduling test has the following form:

$$TimeOfAppTasks + \sum_{o=(sn,rc)} \sum_{j=1}^{M} \sum_{r=1}^{R} e^{Net}_{o,j,r} + \sum_{k=0}^{L} cs^{Net}_k \leq min_{o=(in,out),j=(1,...N)}(p_{o,j})$$

where M is the maximal number of connections, R is the maximal number of tasks in the transport subsystem performed over connection 'j' in direction 'o' (sn/rc) and L is the maximal number of context-switches during the minimal period. 'TimeOfAppTasks' represents the execution time of the tasks and the context-switching times in the application subsystem.

All tasks for 'send' and 'receive' activities in the application and transport subsystem from Figure 12 are globally scheduled as is shown in Figure 14.

2.3.4 Knowledge Requirements

The local schedulability test requires knowledge of the timing constraints of application subsystem tasks. The global schedulability test requires knowledge of the timing constraints of tasks participating in the application and transport subsystem in order to predict the behavior of the end-point system and give QoS guarantees to the users.

To get the required information, we use the QoS parameter database, where

- through *negotiation* and *translation* we get partial information such as the time period of read and write tasks to perform the schedulability tests.

- through *measurements* running before the system is installed and the *installation* of the QoS parameter database we get the processing time of individual tasks as well as context switching time. These values are stored in the QoS parameter database in the system parameter structure (system.application_subsystem.application_tasks and system.transport_subsystem.network_tasks - Figure 7).

2.3.5 Scheduling Implementation

The implementation of the task scheduling will be done under AIX operating system (OS) and its *priority-based scheduler*. This implies a mapping of the tasks scheduled by the mixed scheduling algorithm into proper sequence of priority-based tasks. Further, to avoid unpredictable system activities, proper usage of *priority task assignment* in the OS can provide an approximation of which tasks besides the application and transport subsystem tasks might run without violating the schedulability tests. The implementation of the priority assignment according to the mixed scheduling algorithm, and the implication of the priority-based scheduling on the schedulability tests need further refinement and research.

3 QoS Management in End-Point Architecture - Services and Protocols

The QoS parameter management is an additional component to the transmission and control (call/connection setup) components in the network architecture of the end-points. We propose the end-point architecture which includes *QoS parameter management services* shown in Figure 15.

The main emphasis in this section is on the specification and design of services and protocols which will support the setup of customized call/connections for real-time multimedia applications. Also the transmission protocols at the application and transport subsystem will be discussed.

Our architecture is consistent with the *B-ISDN Protocol Reference Model* (Figure 12.8 in [25]) and Figure 16 shows the QoS components in our extended B-ISDN model. The *higher layers* from the reference model [25] are represented in Figure 16 through the application and transport subsystem. The *control plane* from the reference model [25] is represented in our architecture as call/connection/VC management along the layers and the QoS monitoring functionality. The

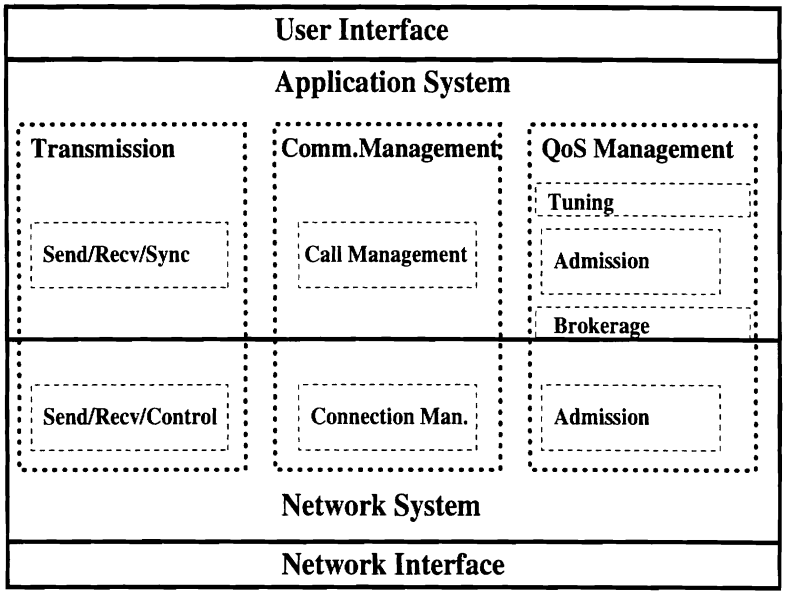


Figure 15: *QoS Parameter Management Services in End-Point Architecture*

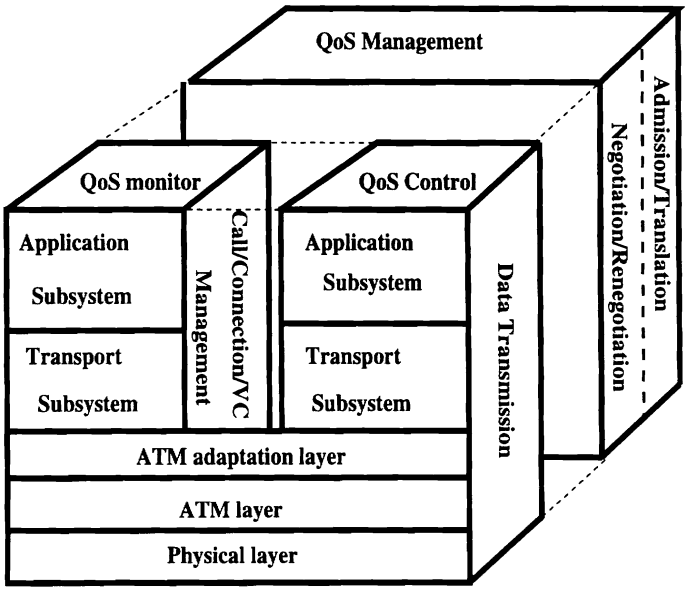


Figure 16: *Extended B-ISDN Reference Model*

user plane from the reference model is implemented through data transmission protocols and QoS control mechanisms. The *management plane* from the reference model [25] is represented in our model through QoS parameter management, where the *plane management* is implemented through peer-to-peer negotiation/renegotiation protocols and *layer management* performs admission control at each layer as well as layer-to-layer communication (Tuning, Brokerage).

3.1 Services

The service entities in the QoS parameter management support the call/connection establishment and work with the QoS parameter database. They include the translation functionality among different parameter sets and tests for the acceptance of the requirements as well as communication capabilities.

3.1.1 Tuning Service

Tuning service provides input of application QoS parameters through human interface as well as presentation of the negotiated application QoS parameters to the human user. The translation of the application QoS parameters is represented through video and audio clips in case of video and audio media, which will run at the negotiated sample rate corresponding to the video frame resolution that the network can support. The tuning service supports tuning of input media as well as output media.

In the case of tuning the input media this action can result in a change of application QoS parameters for input media and a new request on the network for customized connections. In the case of output media only local quality (e.g., sample rate) can be tuned.

This service is present only at the end-point where the human user is the end-user. In the case of robots (machines) this service is not useful.

The tuning service implements layer-to-layer communication between the human user and the application layer. The general state machine for layer-to-layer communication is shown in Figure 17. We support bidirectional negotiation, and therefore k layer can be (in case of human user-application communication) a human user or an application layer and depending on the request direction, the (k-1) layer will be an application if the request came from a human user (k layer) or the (k+1) layer will be a human user if the request came from an application (k layer).

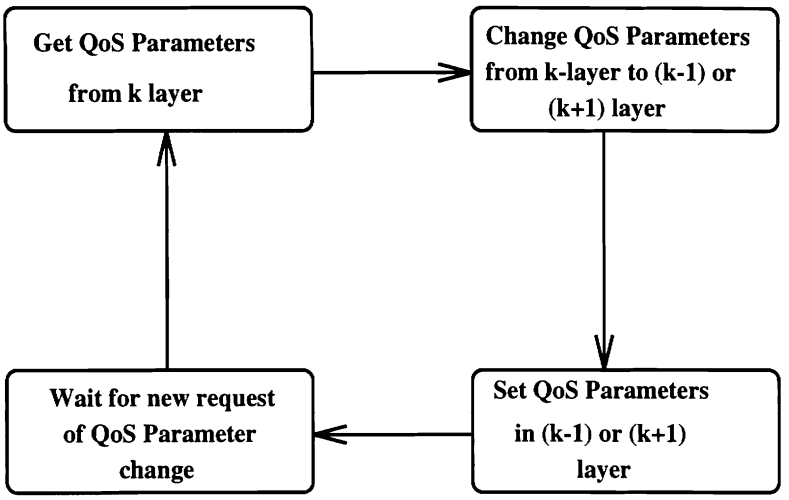


Figure 17: *Layer-to-Layer Communication*

3.1.2 Brokerage Service

The brokerage service implements layer-to-layer communication (Figure 17) between the application and the network layers. This means the brokerage service provides handling of QoS parameters at the application/network interface during the call/connection set up. The *broker* is the entity responsible for the service. The state machine how the broker works is shown in Figure 18.

The functionality of the broker is:

- a *translation process* between application QoS and network QoS, shown in Figure 19, which can include an *integration (multiplexing)* operation (Figure 21) of QoS parameters. Integration occurs when two media with similar application QoS are transported through one network connection. The actual translation process does translation of the application QoS parameters, which might characterize 'N' media, into 'M' network connections where $M \leq N$. Example translation is shown in Figure 20. For translation and integration equations in Figures 20 and 21, we use the notation introduced in section 2.3.1. The media quality parameters are specified as deterministic average bounds (single value) and translated into deterministic average bounds of the connection quality parameters (Figure 20). In the case of QoS integration, even if the media quality parameters of media 1 and media 2 are specified with single values, the resulting media quality parameters might have pair or interval representation of their deterministic bounds (Figure 21).

- a *communication process* to network connection management with *initiation* or *change* of a parameterized connection set up (e.g., in case of initiation: send 'connect request' with

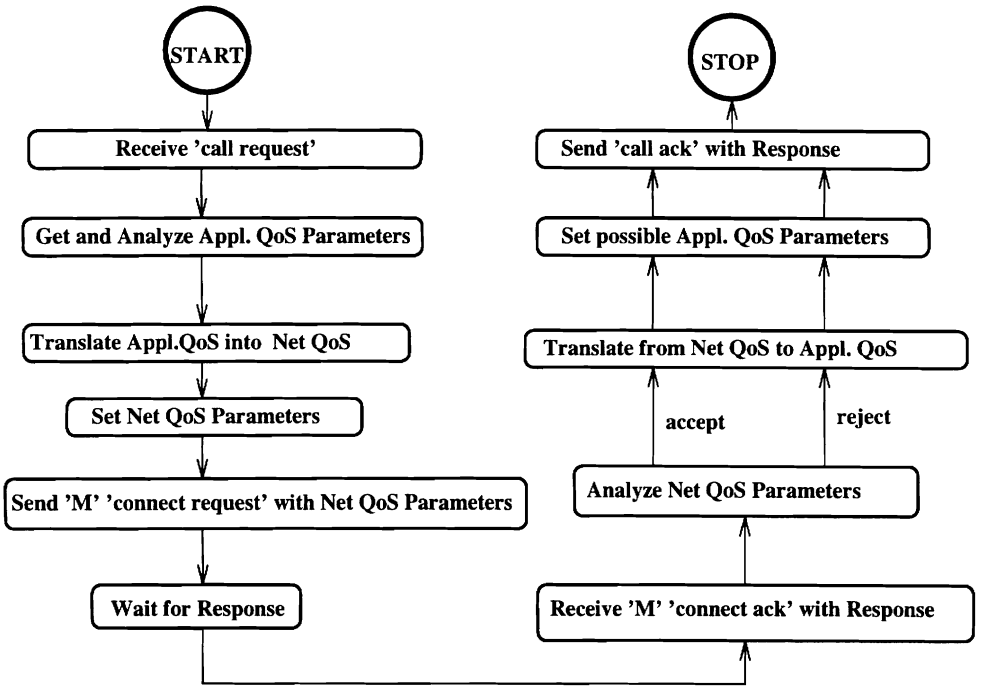


Figure 18: *Brokerage Service - State Machine*

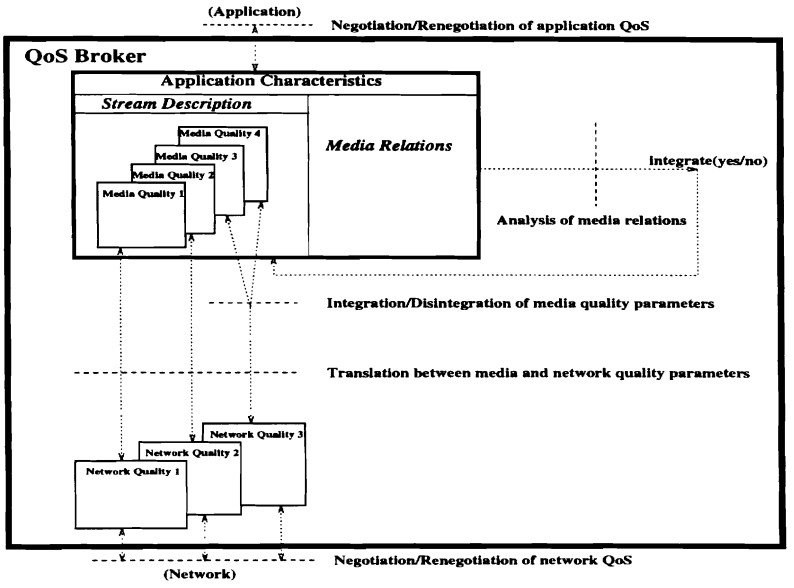


Figure 19: *Translation/Integration Process in QoS Broker*

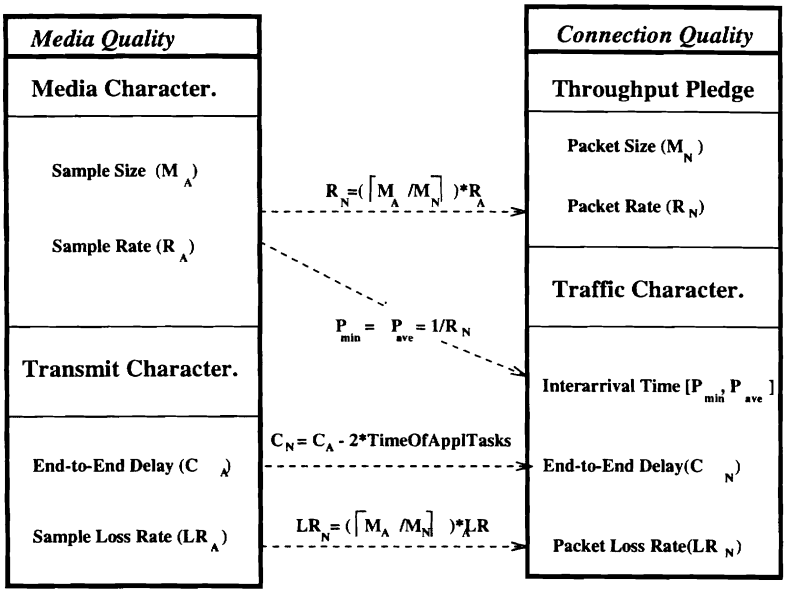


Figure 20: *QoS Parameter Translation (Example)*

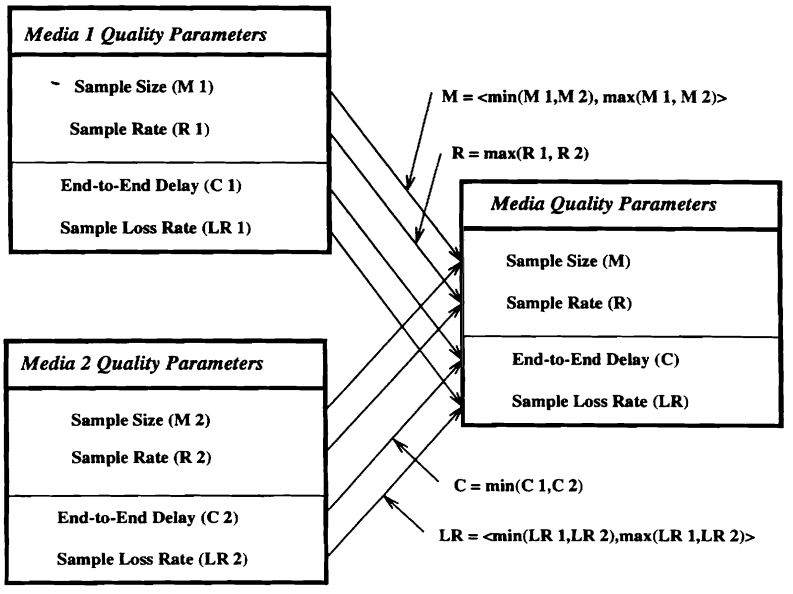


Figure 21: *QoS Parameter Integration (Example)*

network QoS parameters and wait for 'connect ack' with possible network QoS parameters), and

- a *communication process* to application call management with *initiation* or *change* of a customized call set up (e.g., in case of initiation: receive 'call request' with application QoS parameters and send 'call ack' with response and possible application QoS parameters).

The translation has some specific properties:

- The input from an application subsystem are application QoS parameters which include a description of the complete application with all input and output media.
- The broker takes the input medium by medium and maps the media quality parameters with the support of media relations into the outgoing network connection qualities. This means if there are 'N' media which will be sent to remote users, the broker does $M \leq N$ translations into network QoS parameters (Figure 19).
- When the result comes from the network about the outgoing network connections, the broker waits for all M responses and then translates back to application QoS parameters, so that the user can get a global picture of which media in the outgoing call can be supported with what level of quality and make the proper decision about quality degradation if necessary.
- The broker is not involved in handling QoS parameters for incoming (receive) connections. The reason is that we decoupled the negotiation of application QoS and network QoS parameters.

3.1.3 Admission Service

Admission service is an important service to check system resources along the path from the sender to the receiver during the call/connection establishment phase. Admission service has two main jobs:

- *map* between the application/network QoS parameters (Figure 5,6) and the system parameters (Figure 7);
- *check* the availability of resources (e.g., schedulability test and test on space allocation) and based on these tests provide an answer ('reserve', or 'reject') with possible QoS parameters to the negotiation (call/connection set up) process on the sender side and every intermediate

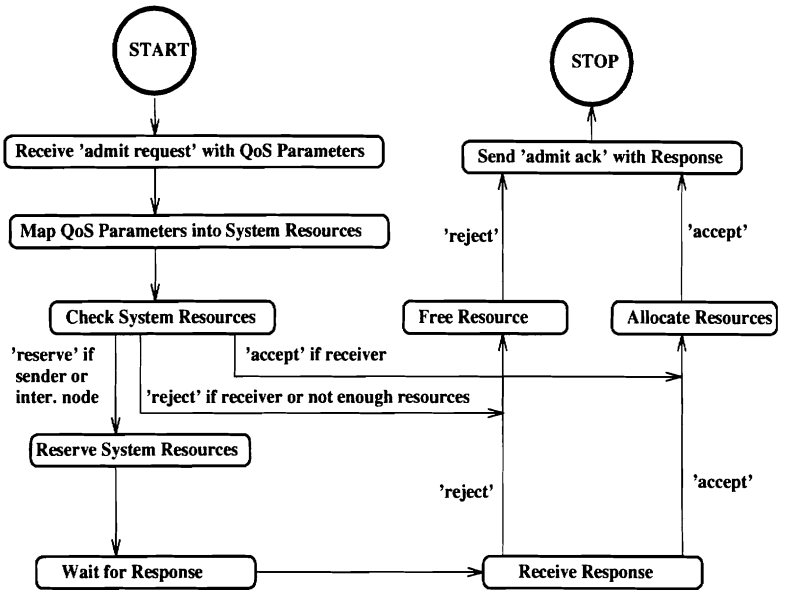


Figure 22: *Admission Service - State Machine*

network node. On the receiver side the admission service provides the answer 'accept' or 'reject' and sends back to the sender the response with possible QoS parameters. The basic state machine of the admission process is shown in Figure 22.

Admission service at the end-point performs at two levels. Once at the application subsystem level to check the user space resources, and second at the transport subsystem level to check the system space resources. At each level, the admission is done for input/outgoing media/connections as well as for output/incoming media/connections. An important admission request is to consider the input/outgoing media/connections and output/incoming media/connections together as far the information is available. The next request for admission service in the transport subsystem is the consideration of application tasks and their processing time.

3.2 Protocols

We consider two main functionalities in application and transport protocols : (1) *call/connection setup* functionality with extension of negotiation capability, (2) *multimedia transmission in application protocol* functionality using *real-time transmission* embedded in the real-time transport/network protocol.

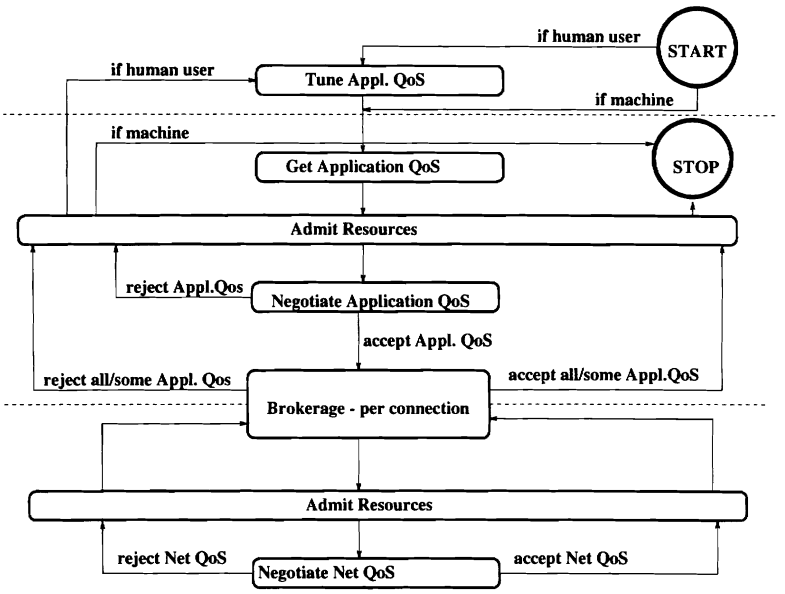


Figure 23: *QoS Parameter Management Services during Call Setup - State Machine*

3.2.1 Call Setup Protocol

The call setup protocol guarantees to set up a customized call between the sender and the receiver based on the application requirements and possible network/system guarantees. The state machine for the call setup with negotiation/admission capabilities is shown in Figure 23.

Call set up protocol includes tasks:

- *set_QoS ("name of QoS parameters")*;

  This functionality is done during the tuning service if the application QoS parameters are specified, or during the brokerage service when the network QoS parameters are obtained (Figure 17).

- *negotiate_peer(input/sending_QoS, remote_address)*;

  This functionality is done by the peer-to-peer negotiation protocol. The negotiation protocol for application QoS parameters is as follows: using a separate control connection (non-parameterized) the application QoS parameters are exchanged. The receiving party checks the incoming multimedia quality and service requirements (e.g., resource /device availability, task/service existence). The result is one of responses 'accept', 'modify', or 'reject'. In the case of a 'modify' response, the receiver must modify its own sensory environment so that it can communicate with the sender. We assume that the sender sends its best possible quality of media and the receiver has to adjust if there are differences. This approach is a scalable

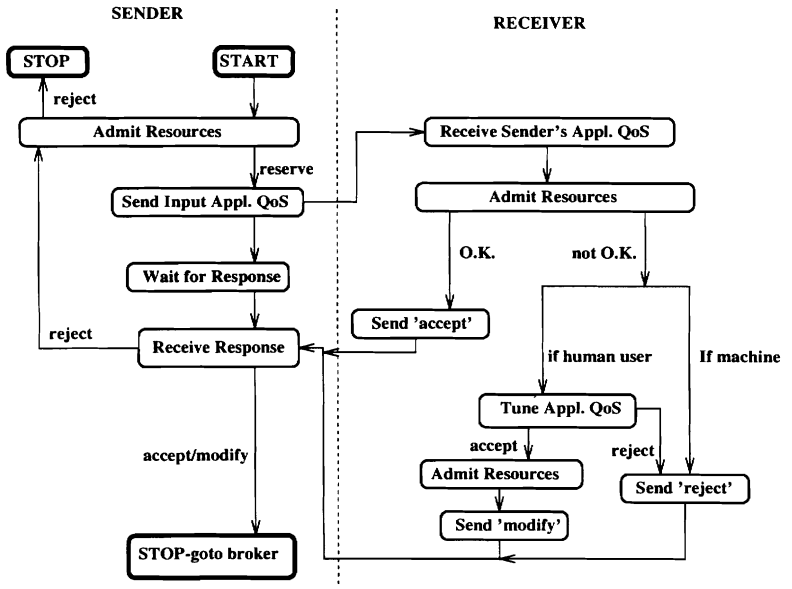


Figure 24: *Negotiation Protocol of Application QoS Parameters - State Machine*

approach especially for large-scale multicasting or broadcasting purposes. The sender reacts to the 'modify' response as it would be 'accept' response. The 'modify' response is important to the receiver for admission of its own resources and for presentation of received media.

The network-to-network negotiation protocol includes results for network QoS acceptance either "accept", or "reject", where the response message includes the possible network QoS parameters that the network can offer and provide. We assume that the resource specification is represented through a deterministic interval of lower and upper bounds of resource values. The 'reject' response is sent if the lower bound of resource request can not be admitted. Otherwise 'accept' response is sent where the upper bound of the resource request can be modified.

The negotiation protocols are shown in Figure 24 and 25.

- *negotiate_layer(input/sending_QoS, remote_address)*;

  This functionality is done by the tuning service if a human user negotiates with the application, or by the brokerage service, if an application negotiates with network (Figure 17).

- *admit_resources(QoS)*;

  This functionality is done by the admission service (Figure 22).

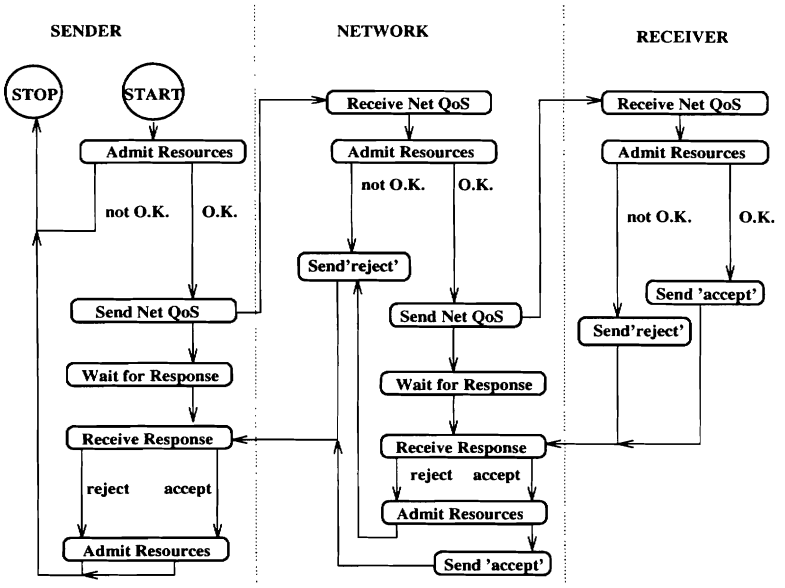


Figure 25: *Negotiation Protocol of Network QoS Parameters - State Machine*

3.2.2 Connection Setup Protocol

A connection set up protocol establishes unidirectional, point-to-point, customized connection. The connection set up request ('connect request') carries negotiation capabilities (fields in the connection set up header for specification of network QoS parameters) which represent the admission reservation request ('admit request'). The connection set up response ('connect ack') includes the negotiation response ('accept' or 'reject') which represents the admission acceptance ('admit ack') of the requested connection. The connection set up protocol in the case of 'accept' is shown in Figure 26.

3.2.3 Application Transmission Protocol

The application on top of the protocol stack invokes the application multimedia protocol [4] for communicating with the remote side.

The tasks and their primitives in the application protocol for support of multimedia transmission are:

- *get_QoS("name of parameter")*
- *read_media(source, sink, sample_size)*

[4] This protocol is a call-oriented protocol with functionalities including call set up, described in 3.2.1, call close, multimedia transmission and embedding negotiation/renegotiation capabilities.

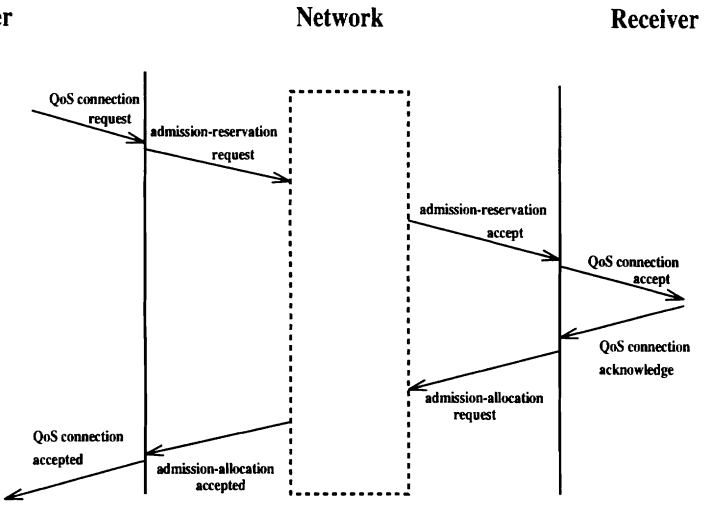


Figure 26: *Connection Set Up Protocol*

- *write_media(source, sink, sample_size)*
- *send_media(source, sink, msg_size)*
- *recv_media(source, sink, msg_size)*
- *set_header(media_type, msg_header)*
- *get_header(media_type,msg_header)*
- *synchronize_media(media_types, msg_headers)*
- *fragment(source, sink, fragment_size, source_pointer)*
- *assemble(source, sink, fragment_size, sink_pointer)*

The sending and receiving activities are generally concurrent activities. Figure 27 shows the state machine of one possible transmission.

3.2.4 Transport/Network Transmission Protocol

The real-time transmission functionality is part of the *Real-Time Transport/Network Protocol* [5]. This functionality includes support of data movement through the end-point such as the *copying*

[5]This protocol is a connection-oriented protocol with primitives for support of connection set up described in section 3.2.2., connection close, and data transmission, including negotiation and renegotiation capabilities.

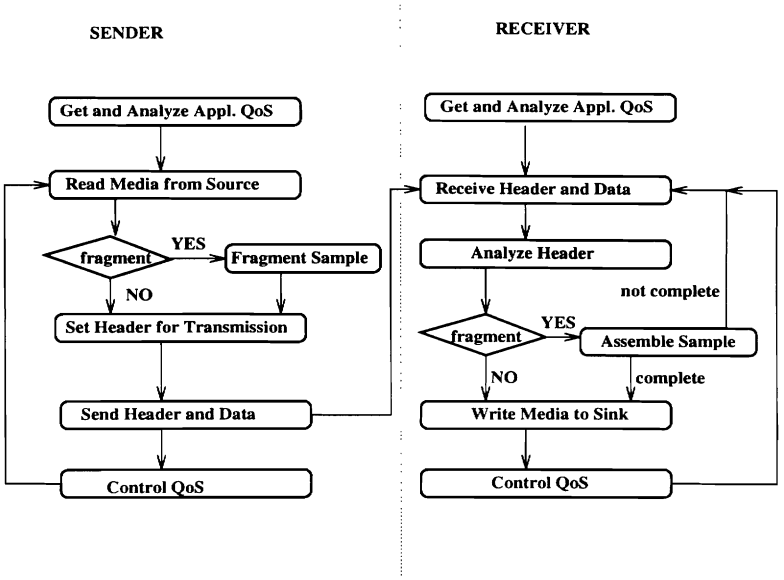


Figure 27: *Media Transmission Protocol (Example) - State Machine*

operation of the data from/to the user space to/from the host interface using the host interface device driver primitives and *rate control* on the sender side. No error control will be included in the transport/network layer because the underlying layer provides checksum for error recognition and reports to the higher layer and we believe this is sufficient right now for the real-time traffic support, although further refinement of the transmission protocol will be studied and implemented.

4 Related Work

4.1 Representation of QoS Parameters

1. **QoS Parameters in Layers**

   Traditional quality of service provided by the network layer per connection and QoS parameters as its representation were considered in the network/transport layer of the OSI architecture [37]. The representation of QoS parameters is two lists of values where the first list gives the goal that the caller wants and the other list gives the minimum value considered as acceptable.

   Current research introduces QoS parameters at every stage of the layered architecture for the high-speed networks. In [20] the QoS parameters were introduced at the ATM and AAL layer, in [32] the QoS parameterization is defined generally for network layers which provide real-time services. The requirements at the application layer are described in [23], [17],[27] in the form of network QoS parameters and in [16] in the form of criteria.

2. **Semantics of QoS Parameters**

Most of the current QoS parameters differ from the parameters used in [37] because of a change in the variety of applications, media sent through networks, and the quality of the networks. While earlier parameters such as *connection establishment delay, connection establishment failure probability, residual error rate, transfer failure probability, resilience connection release delay*, etc. were important to describe and guarantee, today QoS parameters are different.

In [32], the parameters are defined for load and for performance. Network load is characterized through *minimum packet interarrival time* on the channel, the minimum value of the *average packet interarrival time* over an interval of duration I, the *maximum packet size* and the *maximum service time* in the node for the channel's packet. Performance is expressed through the *source-to-destination delay* bound for the channel's packets and the *maximum packet loss rate.*

Jung and Seret [20] differentiate also between the QoS parameters and network performance parameters as bit error ratio, cell error ratio, cell loss ratio, cell transfer capacity, cell insertion rate, delay and jitter.

Generally the QoS parameters meet in their description at the transport and lower layers such as as latency, bandwidth, delay jitter [17],[23].

Application requirements are specified in [16] quantitative and qualitative criteria. Quantitative criteria are those which can be evaluated in terms of certain measures as bits per second, number of errors, etc. The parameters are throughput, delay, response time, rate, data corruption. Qualitative criteria specify the expected service rated for session management as interstream synchronization, ordered delivery of data, error tolerance, maximum data unit size, etc.

3. **Representation of QoS Parameter Values**

In [20] deterministic bounds (pair) are put on AAL QoS parameters: frame error ratio, throughput, delay and jitter as well as on ATM QoS parameters: cell loss ratio, average throughput, delay, jitter cell insertion rate and cell discard ratio.

4. **Service Objects with QoS Parameter Description**

In [37] the description is meant to be for services processing a *transport/network connection*, in [32] for a *'real-time' channel* of a packet switched network. QoS parameter representation in [17] is at the *call, connection, and VC* control level. The Application Programming

Interface (API) of Touring Machine [23] provides a parameterized interface with specification of *connectors* and *ports*. Partridge [27] defines *flows* through QoS parameters as token bucket size and rate, loss and corruption rate, minimum and maximum transit delay and how strongly these quantities have to be guaranteed. At the AAL layer the *service classes* are parameterized. Classification of services according to QoS characterization can be found in [16], [34].

4.2 Communication of QoS Parameters

1. **Negotiation/Renegotiation of QoS Parameters**

   In [37] an option for negotiation is introduced. This means during the connection establishment the QoS parameters are exchanged through N-CONNECT primitives and the negotiated parameters are taken during the transmission phase. If the network service is unable to provide at least the minimum value of any parameter specified either by caller or callee, the connection establishment fails.

   Negotiation in OSI architecture is possible only between the peer layers, not between the consecutive layers. The specification of the negotiation capabilities in the transport layer is described in the framework of the OSI 95 Transport Service [8]. In [17] the CCITT-I Series perspective is presented.

   A signaling scheme from user to the ATM network is described in [17], [26]. The signaling protocol EXPANSE [23] supports the establishment and modification of complex multimedia connections for B-ISDN.

   There is no renegotiation in CCITT ATM Recommendations ([17]). Renegotiation is considered in signaling protocol EXPANSE [23], Lancaster transport protocol [28], [18].

2. **Translation of QoS Parameters**

   Translation between AAL and ATM layer QoS parameters is described in [20], between the application requirements and network QoS in [21].

4.3 QoS Guarantees

An overview of different approaches (tightly controlled, approximate, bounding or observation-based approaches) towards provision of QoS guarantees can be found in [19]. The research in this area goes towards *adaptive admission control* of network QoS parameters, and *real-time scheduling* in OS for support of resource management. Several adaptive schemes using negotiation and

notification in the case of service degradation are presented in the literature. The work on QoS parameter monitoring and policing is rather scarce.

1. **Admission Service**

   The admission service maps the QoS parameters towards protocol functions (scheduling, flow control, error control, sequence control, etc.) which then provide the QoS guarantees in the transmission protocols. Admission service at the cell level is presented in [14], [27]. In [16] a mapping between service classes and protocol functions was given. Mapping between layer, QoS parameters and protocol function (e.g., at the application layer in order to bound jitter, delay, and throughput we need scheduling) is described in [17].

2. **Scheduling and other QoS Parameter Control Functions**

   There is an extensive work on

   - real-time scheduling (deadline based scheduling, rate monotone scheduling, etc.) for QoS guarantees [32],[14],[29], [27],
   - flow control (rate based) [32], and
   - error control (forward error correction) [12].

   Some work is done in specification and design of adaptive real-time resource management in order to support digital multimedia services [1], [2].

3. **QoS Monitoring and Policing**

   In order to guarantee QoS, we need not only admission control, but also monitoring and policing of QoS parameters during the transmission [17].

4.4 QoS Support in End-Point Architectures

The QoS support is embedded in different components of the network architecture. We will consider only related work for end-point architectures. The OSI architecture provides QoS in the network layer (in the connection set up protocol) and enhancements in the transport layer [37], [8].

In [35] the end-points have only three layers (application, orchestration and multimedia mechanism layers). The QoS handling is embedded in the multimedia mechanism layer, and the QoS description is done in DSL (Device Specification Language) language at the application layer. The QoS description applies to the DSL *streams*.

Most architectures at the end-points consider the QoS handling only in transport subsystem layers [15],[16], [9], [13]. Current research in [17], [2] also goes in the direction of a provision of a QoS architecture which considers different protocol functions for QoS guarantees in different layers.

In the Lancaster QoS architecture [17] the application layer includes scheduling in order to satisfy jitter, delay and throughput. In the orchestration layer one needs jitter provision for jitter correction, rate regulation for throughput, and for multiple connection synchronization. In the transport subsystem the throughput guarantee is done by flow control, the error control needs FEC (Forward Error Correction). An ATM network is assumed to have scheduling for jitter, delay, and throughput guarantees.

Jung and Seret [20] also consider a QoS framework with respect to the QoS provision in different layers at the end-points. Their architecture includes user, higher layers, lower layers at the end-point, then user/network interface and network.

4.5 Summary of Related Work

The overview of the QoS work in this section shows that there are research results which partially address the QoS approach at the transport and network layers; most of the work is concentrated on QoS parameter control (e.g., scheduling) in these layers. The proposed algorithms for QoS parameter control are tested through different ATM network simulation scenarios, or a TCP/IP protocol stack, running over Internet, is modified embedding new QoS control mechanisms [5], [6], [7].

Experiments with ATM networks are needed, but experimental ATM high-speed networks are still not widespread. Applications are simulated as origins and destinations of certain traffic behavior, and the applications are not embedded in the protocol stack at the end-points.

Early ATM network testbeds such as AURORA [30] and XUNET [31] are providing the experimental environment for implementation of

- customized calls and connections,
- QoS handling in the end-point architectures, and
- OS support for QoS handling in the end-points.

5 Application to Telerobotics

We are exploring our approach in the context of an actual application, that of telerobotics and digital teleoperation [24].

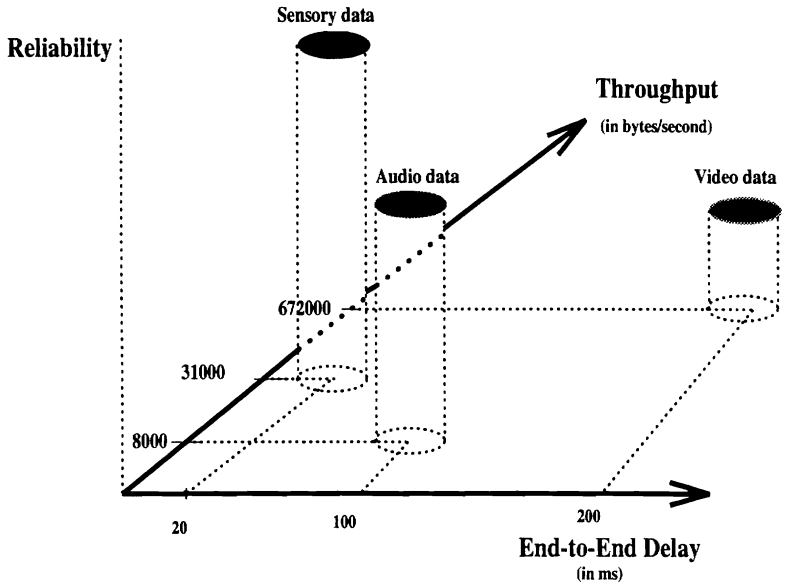


Figure 28: *Network QoS Parameter Space in Telerobotics*

5.1 Why Teleoperation

The telerobotics/teleoperation application is a non-trivial application and provides many challenges to network research. The reason is that the involved media *sensory data*, *audio*, and *video* with their complex application requirements fill different portions of the network QoS parameter space as it is shown in Figure 28. The medium such as video puts high requirement and therefore constraint on the network throughput but the reliability request as well as the end-to-end delay request are more relaxed than it is in the case of the audio or sensory data. On the other hand the medium such as sensory data requires high reliability and strict constraint on end-to-end delay from the network during its transmission but the throughput requirement is low.

Teleoperation allows a remote operator to exert force or to impart motion to a slave manipulator. The operator can also experience the force and resulting motion of the slave manipulator, known as "kinesthetic feedback". An operator is also provided with visual feedback and possibly audio feedback as well.

Visual information requires at least megabit bandwidth with frame rates in excess of ten frames per second. Normally, teleoperation makes use of two to three video channels. The kinesthetic communications channel is required in both directions for each manipulator. Kinesthetic channels require transmission of a few hundreds of bits per sample, at about a 50-500Hz rate. As one might expect, there are strict timing requirements on manipulator channels ("robotics data") - irregular or missing data can result in physical damage. Along with these channels might be channels for

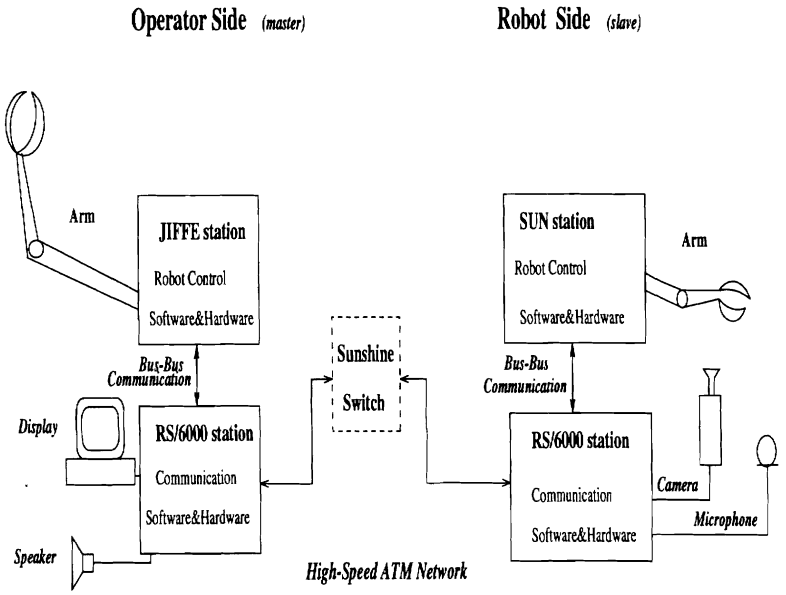


Figure 29: *Intended System Configuration*

audio and video information. The challenge is the potential conflict of QoS requirements when resources are limited.

This application has dynamic changes in its requirements over its execution because the physical information changes, the robot arms are mobile, they may obscure or expose a camera while moving. The changes of the physical information may result in renegotiation of requirements among the remote sites as well as changes in network guarantees. The complex timing requirements of the telerobotics application gives us a realistic and demanding platform with which we can study parameterized call/connection management and negotiation. Telerobotics employs distributed control and execution mechanisms which force some real-time requirements on the network.

5.2 System Configuration

Our test system configuration is shown in Figure 29.

The network solution employs a point-to-point link to a high-speed ATM switch to form an ATM LAN. The LAN is interconnected to WAN facilities through the same switch; links are fibers operating with SONET OC-3c or OC-12. We operate ATM over SONET OC-3c, giving a bandwidth of 155 Mbps. ATM over SONET provides the key features of low error rates, high-bandwidth, and facilities which can be used for paced data delivery.

The communication software and hardware support for video, audio and ATM host interface are and will be implemented on IBM RISC System/6000 workstations using AIX operating system. To obtain robotics sensory data over the ATM network we connect the SUN-4 and RS/6000 stations

with an S bus-to-Microchannel bus interconnection card at the slave side. On the master side a real-time processor (called "JIFFE" and labeled as such in Figure 29) and a dedicated IBM PC provide the robot control. A BIT3 bus connector card connects the JIFFE processor with the IBM RS/6000 workstation.

6 Expected Results in the Thesis Work

Some of the major results expected from the thesis work include:

- A practical demonstration that an application with strict Quality of Service requirements can be accommodated within cell-switched network technology operating at high-speed.
- A breakdown of the important elements in architectures for supporting such applications, such as schedulers and priority services.
- An initial understanding of the limitations on flexibility and performance imposed by application requirements. For example, we would expect a tradeoff between tightness of message interarrival delays and the number of channels supported with such delays. These limitations and their relationship would be represented as curves, derived from experimental measurements. We expect that some functions approximating the measured results can be used to predict results outside of our measurement capabilities.
- We hope, also, to be able to give prescriptive advice for the design of new systems, e.g., with more available scheduling control, to remedy problems we encounter. There may also be implications for workstation hardware design.

7 Plan of Steps and Measurements in the Thesis Work

The refinment of the QoS framework as well as the implementation and evaluation of the experiment will be done in several steps:

1. Determine limits on schedulability in AIX OS to support the practical boundaries on measurement;
2. Refine admission services and call/connection establishment protocols;
3. Prototype call setup with QoS parameter management support;
4. Refine control software (transmission protocol) for host interface in loop back configuration;

5. Do end-to-end measurements on idle dedicated ATM link and refine Real-time Transport/Network Protocol (RTNP);

6. Prototype sender-receiver communication for robot control, audio and video data;

7. Add load to dedicate link by loading of other VC's at the end-point and ATM switch.

8. Prototype renegotiation of QoS parameters.

9. Do evaluation of the implementation, i.e. graph results versus:

   - number of processes;
   - number of processes in various priorities;
   - computation of feasible schedule according to admitted media/connections (using schedulability tests) and its accuracy during the transmission;
   - minimal time for negotiation/renegotiation;
   - priority versus jitter;
   - unpredictable system interactions versus QoS requirements;

References

[1] Michael B. Jones, "Adaptive Real-Time Resource Management Supporting Modular Composition of Digital Multimedia Services", *NOSSDAV '93 Workshop Proceedings*, Lancaster, England, 3-5 November, 1993

[2] Andrew Campbell, Geoff Coulson and David Hutchison, "A Multimedia Enhanced Transport Service in a Quality of Service Architecture", *NOSSDAV '93 Workshop Proceedings*, Lancaster, England, 3-5 November, 1993

[3] K.K. Ramakrishnan, Lev Vaitzblit, Cary Gray, Uresh Vahalia, Dennis Ting, Percy Tzelnic, Steve Glaser, Wayne Duso, "Operating System Support for a Video-On-Demand File Service", *NOSSDAV '93 Workshop Proceedings*, Lancaster, England, 3-5 November, 1993

[4] Craig Partridge, "Gigabit Networking", *Textbook*, published by Addison-Wesley Publishing Company, October 1993

[5] Lixia Zhang, Bob Braden, Deborah Estrin, Shai Herzog, Sugih Jamin, "Resource ReSerVation Protocol (RSVP)", *Internet Draft*, October 1993

[6] Scott Shenker, David D. Clark, Lixia Zhang, "Service Model for an Integrated Services Internet", *Internet Draft*, October 1993

[7] Bob Braden, Dave Clark, Scott Shenker, "Intergated Services in the Internet Architecture: an Overview", *Internet Draft*, October 1993

[8] Andre Danthine, Oliver Bonaventure, "From Best Effort to Enhanced QoS", *Technical Report*, Nr. R2060/ULg/CIO/DS/P/004/b1, University of Liege, Belgium, July 1993

[9] Frank Oliver Hoffman, Luca Delgrossi, "A Detailed Tour of ST-II for the Heidelberg Transport System", *Technical Report*, Nr. 43.9302, IBM European Networking Center Heidelberg, Germany, 1993

[10] Derick Jordaan, Martin Paterok, Carsten Vogt, "Layered Quality of Service Management in Heterogeneous Networks", *Technical Report*, Nr. 43.9304, IBM European Networking Center Heidelberg, Germany, 1993

[11] Martin DePrycker, "Asynchronous Transfer Mode - Solution for Broadband ISDN", *Textbook*, second edition, published by Ellis Horwood Publisher, 1993

[12] E.W.Biersack, "Performance Evaluation of Forward Error Correction in an ATM Environment", *IEEE Journal on SAC*, May 1993, Vol.11, Nr.4, pp.631-640

[13] D.Feldmeier, "A Framework of Architectural Concepts for High-Speed Communication Systems", *IEEE Journal on SAC*, May 1993, Vol.11, Nr.4, pp.480-488

[14] J.M.Hyman,A.A.Lazar, G.Pacifici "A Separation Principle between Scheduling and Admission Control for Broadband Switching", *IEEE Journal on SAC*, May 1993, Vol.11,Nr.4, pp.605-616

[15] D.C.Schmidt, Tatsuya Suda, "Transport System Architecture Services for High-Performance Communications Systems", *IEEE Journal on SAC*, May 1993, Vol.11, Nr.4, pp.489-506

[16] M.Zitterbart, B.Stiller, A.N.Tantawy, "A Model for Flexible High-Performance Communication Subsystems", *IEEE Journal on SAC*, May 1993, Vol.11, Nr.4, pp.507-517

[17] A.Campbell, G.Coulson, G.Garcia, D.Hutchison, H.Leopold, "Integrated Quality of Service for Multimedia Communications", *IEEE INFOCOM'93 Conference Proceedings*, Volume II, San Francisco, CA, March 1993

[18] S.Crosby, "In-call Renegotiation of Traffic Parameters", *IEEE INFOCOM'93 Conference Proceedings*, Volume II, San Francisco, CA, March 1993

[19] Jim Kurose, "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks", *Computer Communication Review*, January 1993, Vol.23, Nr.1, pp. 6-15

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

[21] K.Nahrstedt, J.Smith, "Revision of QoS at the Application/Network Interface", *Technical Report*, Philadelphia, PA, January 1993

[22] C. Brendan S. Traw, Jonathan M. Smith, "Hardware/Software Organization of a High-Performance ATM Host Interface", *IEEE Journal on Selected Areas in Communications (Special Issue on High Speed Computer/Network Interfaces)* Vol. 11, No. 2, February 1993, pp.240-253

[23] Bellcore Information Networking Research Laboratory, "The Touring Machine System", *Communications of the ACM*, January 1993, Vol.36, Nr.1, pp.68-77

[24] Ruzena Bajcsy, David J. Farber, Richard P. Paul, Jonathan M. Smith, "Gigabit Telerobotics: Applying Advanced Information Infrastructure", *Technical Report*, MS-CIS-93-11, University of Pennsylvania, Philadelphia, PA, January 1993

[25] William Stallings, "ISDN and Broadband ISDN", *Textbook*, second edition, published by Macmillan Publishing Company, 1992

[26] D.Ferrari, J.Ramaekers, G.Ventre, "Client-Network Interactions in Quality of Service Communication Environments", *Proc. 4th IFIP Conf. on High Performance Networking*, pp. E1-1E1-14, Liege, Belgium, December 1992

[27] S. Keshav, "Report on Workshop on QoS Issues in High-Speed Networks", *Computer Communication Review*, October 1992, Vol.22, No.5, pp.74-85

[28] Andrew Campbell, Geoffrey Coulson, Francisco Garcia, David Hutchison, "A Continuous Media Transport and Orchestration Service", *SIGCOMM'92 Conference Proceedings*, Baltimore, Maryland, August 17-20, 1992

[29] 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

[30] D. Clark, D. Tennenhouse, D. Farber, J. Smith, B. Davie, W. Sinkoskie, I. Gopal, B. Kadaba, "An Overview of the AURORA Gigabit Testbed", *IEEE INFOCOM '92 Conference Proceedings*, Volume II, Florence, Italy, 6-8 May, 1992,

[31] A. Fraser, C. Kalmanek, A. Kaplan, W. Marshall, R. Restrick, "Xunet 2: A Nationwide Testbed in High-Speed Networking", *IEEE INFOCOM '92 Conference Proceedings*, Volume II, Florence, Italy, 6-8 May, 1992

[32] D.Ferrari, D.C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE Journal on SAC*, Vol.8,Nr.3, April 1990, pp. 368-379

[33] Dinesh C. Verma "Quality of Service in ATM Networks", *Technical Report*, University of California, Berkeley, CA, 1990

[34] A.A.Lazar, G.Pacifici, J.S. White, "Real-Time Traffic Measurements on MAGNET-II", *IEEE Journal on SAC*, Vol.8, Nr.3, April 1990, pp. 467-483

[35] C.Nicolau, "An Architecture for Real-Time Multimedia Communication Systems" *IEEE Journal on SAC*, Vol.8., Nr.3, April 1990, pp. 391-400

[36] "Real-Time Computing Systems: The Next Generation", edited by John A. Stankovic, *Tutorial Hard Real-Time Systems*, published by Computer Society Press of the IEEE, 1988

[37] A.S. Tannenbaum, "Computer Networks", *Textbook*, second edition, published by Prentice Hall, 1988

[38] R.M. Metcalfe, D.R. Boggs, "ETHERNET: Distributed Packet Switching For Local Computer Networks", *Communications of the ACM*, July 1976, pp. 395-404

[39] V.G. Cerf, R.E. Kahn, "A Protocol for Packet Network Intercommunication", *IEEE Transaction on Communications*, Vol. COM-22, May 1974, pp. 637-648

[40] 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