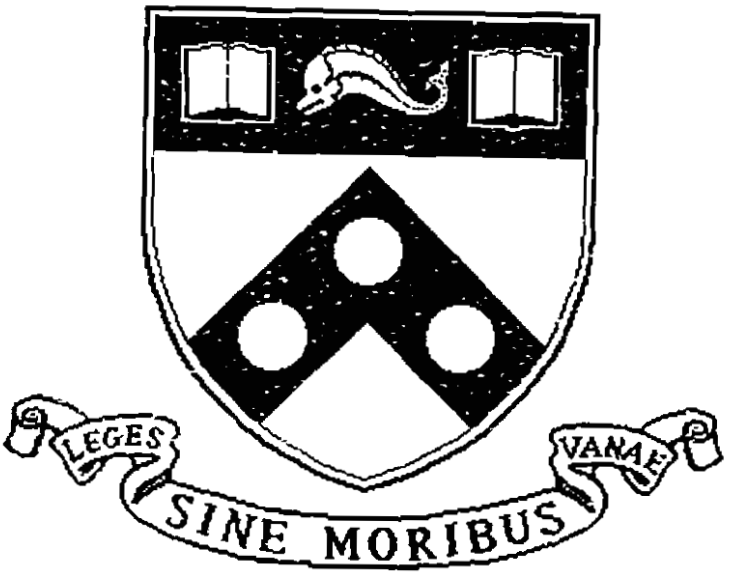
and: submitted to IEEE Computer, Special Issue on Multimedia, April 95 (requested by the editors)

Resource Management in Multimedia Networked Systems

MS-CIS-94-29
DISTRIBUTED SYSTEMS LAB 79

Klara Nahrstedt
Ralf Steinmetz



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

May 1994

[NaSt94] *Klara Nahrstedt, Ralf Steinmetz;* **Resource Management in Multimedia Networked Systems;** Vorversion des eingeladenen Beitrags eingereicht bei IEEE Computer, 1994 [NaSt95]; Technical Report, University of Pennsylvania, PA, USA, 1994.

Resource Management in Multimedia Networked Systems

Klara Nahrstedt
Distributed System Laboratory
University of Pennsylvania
200 South 33rd Street
Philadephia, PA 19104, USA

klara@aurora.cis.upenn.edu
Tel: (215) 573 3639
Fax: (215) 573 2232

Ralf Steinmetz
IBM European Networking Center
Vangerowstrasse 18
D-69115 Heidelberg
Germany

steinmet@dhdibmip.bitnet
Tel: +49 6221 594280
Fax: +49 6221 59 3400

Abstract

Error-free multimedia data processing and communication includes providing guaranteed services such as the colloquial telephone. A set of problems have to be solved and handled in the *control-management* level of the host and underlying network architectures. We discuss in this paper 'resource management' at the host and network level, and their cooperation to achieve global guaranteed transmission and presentation services, which means end-to-end guarantees. The emphasize is on 'network resources' (e.g., bandwidth, buffer space) and 'host resources' (e.g., CPU processing time) which need to be controlled in order to satisfy the *Quality of Service (QoS)* requirements set by the users of the multimedia networked system.

The control of the specified resources involves three actions: (1) properly *allocate* resources (end-to-end) during the multimedia call establishment, so that traffic can flow according to the QoS specification;

(2) *control* resource allocation during the multimedia transmission; (3) *adapt* to changes when degradation of system components occurs. These actions imply the necessity of: (a) new services, such as admission services, at the hosts and intermediate network nodes; (b) new protocols for establishing connections which satisfy QoS requirements along the path from sender to receiver(s), such as resource reservation protocol; (c) new control algorithms for delay, rate, and error control; (d) new resource monitoring protocols for reporting system changes, such as resource administration protocol; (e) new adaptive schemes for dynamic resource allocation to respond to system changes; and (f) new architectures at the hosts and switches to accommodate the resource management entities.

This article gives an overview of services, mechanisms and protocols for resource management as outlined above.

Keywords: multimedia, multimedia networking, resource management, guaranteed service, multimedia protocols, quality of services, multimedia systems

1 Introduction

With the advent of multimedia in computing and communications, a new set of requirements has been imposed on related components: (1) *Fast Transfer* – High-speed data transfer [5] is a key issue which allows compressed audio and video data to be moved in local environments as well as over networks; (2) *Guaranteed Delivery* – Multimedia applications demand processing of audio and video data such that humans perceive these media in a natural and error-free way. This continuous-media data has its origin in sources like microphones, cameras and files. From these sources, the data is transferred to destinations such as speakers, video windows and files, located at the same computer or at a remote station. On the way from source to destination, the original data is processed by at least some type of move, copy, or transmit operation. Hence, it is not sufficient to transfer data in a fast mode; data delivery must be *guaranteed.* Further, guaranteed delivery protects the users from two sources of variability: *misbehaved users* and *network load fluctuation.* Misbehaved users can, for example, send packets at a higher rate than the bandwidth allocated to them and network load fluctuation may cause a higher arrival rate from a channel at some switch, even though the channel satisfies the bandwidth allocation constraint at the entrance to the network.

Until now multimedia networked applications only received from the hosts and networks *best-effort* delivery, provided by best-effort services. In order to provide *guaranteed services*, several new problems have to be solved and handled in the control-management level of the host and underlying network architectures. We concentrate on *resource management* at the hosts and networks, and their cooperation to achieve guaranteed services for multimedia in end-to-end fashion. The emphasis is on *network resources* (e.g., bandwidth, buffer space) and *host resources* (e.g., CPU processing time) which need to be controlled in order to satisfy the *Quality of Service (QoS)* requirements set by the users of the *Multimedia Networked System (MNS)* and, therefore, to achieve delivery guarantees.

Guaranteed services imply three main actions for resource management: (1) *reserve* and *allocate* resources (end-to-end) during the multimedia call establishment so that traffic can flow according to the QoS specification. This implies distribution (negotiation) of the QoS specification to every system component of the MNS; (2) *provide* resources according to the QoS specification, which means controlling resource allocation during multimedia delivery using proper service disciplines; and (3) *adapt* to resource changes during an on-going multimedia delivery.

This paper is a survey of the whole scope of resource management for multimedia computing and communication; hence, it ties together available results in various domains and builds bridges where

a relationship is required. We do not present a single focused point of view, but rather provide the whole picture. We begin with a short survey on multimedia requirements leading to the notion of quality of service and resources (Section 2). Section 3 presents the required concepts necessary for the multimedia call set-up. Enforcement of QoS during a multimedia call is described in Section 4. During such a call, the QoS may change (decrease or increase), resulting in resource modification. The corresponding schemes and concepts to accommodate these changes are presented in Section 5. In the conclusion we briefly describe some implications for system architectures.

2 Requirements of Multimedia Networked Systems and their Implications

Distributed multimedia applications put new requirements on application designers as well as network protocol and system designers. We will analyze the most important enforced requirements with respect to resource management and their implications.

2.1 User and Application Requirements

New enabling technologies, such as fiber-optic networks and digital audio/video support for computers, increased an emphasis on distributed multimedia applications (e.g., multimedia mail, conferencing, screen sharing, or virtual desk-tops). These networked multimedia applications impose new requirements on data handling in computing and communications because they need (1) a *substantial data throughput*, (2) *fast data forwarding*, and (3) *service guarantees*.

Audio and video data have a stream like behavior, and they demand, even in a compressed mode, high *data throughput*. As an example let us consider MPEG-2 compressed audio and video data. The MPEG-2 standards define various compression schemes for video with associated audio. There are specified three audio layers with different implementations and quality requirements. Video is arranged in a set of profiles and layers which correspond to different image qualities and sizes. The resulting data streams range from 4 Mbit/s up to 100 Mbit/s. An excellent quality, compared to today's television, can already be achieved with about 4 Mbit/s. In a workstation or network, several streams may exist concurrently, demanding a high throughput. Further, the data movement requirements on the local end-system translate into terms of manipulation of large quantities of data in real-time where, for example, data copying can create a bottleneck in the system.

Fast data forwarding imposes a problem on the end-systems where these different applications can exist in the same end-system and may have requirements on data movement ranging from nor-

mal error-free data transmission to new time-constrained types of traffic. This requirement implies careful spatial and temporal resource management in the end-systems and routers/switches. The application imposes constraints on the total maximal end-to-end delay. In a retrieval-application, such as video on demand, a delay of up to 1 sec may be easily tolerated. On the other hand, dialogue-applications, such as a videophone or videoconference, demand end-to-end delays lower than 200 msec in order to allow a natural communication between the users.

The distributed multimedia applications need *service guarantees*, otherwise they will not be accepted as these systems compete against radio and television services. In order to achieve services guarantees, resource management must be used. Without resource management in end-systems and switches/routers, multimedia systems cannot provide reliable QoS to the users because transmission over unreserved resources leads to dropped or delayed packets, violating requirements.

Further, several specific demands which ease the implementation must be taken into account. One example is multicast, which is important for many multimedia distributed applications in terms of sharing resources, like the network bandwidth, and the communication protocol processing at end-systems.

2.2 Processing and Protocol Constraints

The operating system (OS) implementations and the communication protocols have some conflicting constraints which need to be considered when we want to match application requirements and system platforms.

The employment of continuous media in multimedia systems also imposes additional, new requirements on the *end-system architecture*. A typical multimedia application does not require processing of audio and video to be performed by the application itself, although it can, as it is done in MIT's VuNet, implemented by Tennenhouse's group. Usually data are obtained from a source (e.g., microphone, camera, disc, network) and are forwarded to a sink (e.g., speaker, display, network). In such a case, the requirements of continuous-media data are satisfied best if they take 'the shortest possible path' through the system, i.e., copy data directly from adapter to adapter [5]. The application program then merely sets the correct switches for the data flow by connecting sources to sinks. Hence, the application itself never really touches the data as it is the case in traditional processing. In multimedia systems such an adapter to adapter connection is defined by the capabilities of the two involved adapters and the bus performance. In todays systems this connection is static. A problem with direct copying from adapter to adapter is the control and the change of quality of service parameters.

The architecture of the protocol processing system is another issue to be considered in the system architecture. Protocols involve considerable *data movement* because of the layered structure of the communication architecture. However, copying data is expensive and has become a bottleneck; hence other mechanisms have to be found. An appropriate method of moving data can be found in the *paradigm of streaming* from source to destination. This means that the multimedia application opens devices, establishes a connection between them, starts the data flow, and returns to other duties. This architecture of low-level data streaming corresponds with proposals for using additional new busses for audio and video transfer within a computer. It also enables a switch-based rather than a bus-based data transfer architecture [1].

Different layers of the communication system may have different Protocol Data Unit (PDU) sizes, hence *segmentation* and *reassembly* of the PDU occur. These functions must be performed quickly and efficiently. Thus, these functions of protocol stack, at least in the lower layers of the communication system, are implemented in hardware or through efficient mechanisms in the software [5].

Some parts of protocols may use a *retransmission* mechanism to achieve a reliable data delivery, which imposes requirements on buffer space for queues at the expense of larger end-to-end delays.

The new underlying packet/cell networks, which work in an *Asynchronous Transfer Mode (ATM)* (although, for example, FDDI also offers an isochronous transfer mode which is best suited for multimedia transmission), put requirements on the protocol design for continuous media. The higher protocols have to provide synchronous behavior to the application, but they rely on an asynchronous behavior of the service provider at the packet/cell level. This means introduction of connection-oriented protocols where, during the connection establishment, preparation for synchronous transmission of continuous media has to occur [6].

2.3 Quality of Service

The user/application requirements on the MNS are mapped into services which make the effort to satisfy the requirements. Because of the heterogeneity of the requirements, coming from different distributed multimedia applications, the services in the multimedia systems need to be parameterized. Parameterization allows for flexibility and customization of the services, so that each application does not result in implementing of a new set of service providers.

Parameterization of the services is defined in ISO (International Standard Organization) standards through the notion of *Quality of Service (QoS)*. The ISO standard defines QoS as a concept for specifying how 'good' the offered networking services are. QoS can be characterized by a number

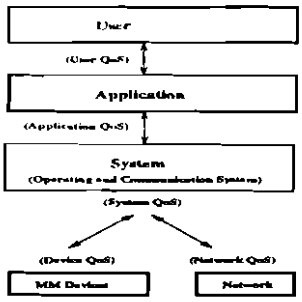


Figure 1: *Architectural View of a Multimedia Networked System*

of specific parameters. There are several important issues which need to be considered with respect to QoS.

2.3.1 Layering of MNS

Traditional QoS (ISO standards) was provided by the network layer of the communication system. An enhancement of QoS was achieved through inducing QoS into transport services. For MNS, the QoS notion has to be extended because many other services contribute to the end-to-end service quality. In order to discuss further QoS and resource management, we need a layered model of the MNS. We assume throughout this the paper the model shown in Figure 1. The MNS consists of three layers: ***application, system*** (including communication services and operating system services), and ***devices*** (network and multimedia (MM) devices). Above the application may or may not reside a human user. This implies the introduction of QoS in the application (application QoS)[1], system (system QoS), and network (network QoS). We will concentrate on the network device and its QoS because we are interested in MNS. The MM devices find their representation (partially) in application QoS.

2.3.2 Service Objects

Services are performed on different objects, for example, media sources, media sinks, connections, virtual circuits (VCs), hence the QoS parameterization specifies these service objects. In ISO standards, the QoS description is meant to be for services, processing a *transport/network connection.* In [18], the services operate over a *'real-time' channel* of a packet switched network. In [3], a QoS parameter specification is given for *call, connection, and VC* objects. In RSVP (Resource Reservation Protocol) a flow specification is given [9] for parameterization of the *packet scheduling*

[1] In the case of having a human user, the MNS may also have an user QoS specification.

mechanism in the routers or hosts. At higher layers in communication systems, the service objects, for example, *media* [2], or *streams* [7] may be specified.

2.3.3 QoS Parameters

The set of chosen parameters for the particular service determines what will be measured as the QoS. Most of the current QoS parameters differ from the parameters described in ISO because of the variety of applications, media sent, and the quality of the networks and end systems. This also leads to many different QoS parameterizations in the literature. We give here one possible set of QoS parameters for each layer of MNS, as shown in Figure 1.

The *application QoS parameters* describe requirements for the application services possibly specified in terms of (1) *media quality* which includes the media characteristics and their transmission characteristics, such as end-to-end delay, and (2) *media relations* which specify the relations among media, such as media conversion, or inter/intra stream synchronization [2].

The *system QoS parameters* describe requirements on the communication services and OS services resulting from the application QoS. They may be specified in terms of *quantitative* and *qualitative criteria.* Quantitative criteria are those which can be evaluated in terms of certain measures, such as bits per second, number of errors, task processing time, PDU size, etc. The QoS parameters are throughput, delay, response time, rate, data corruption at the system level, task and buffer specification. Qualitative criteria specify the expected services needed for provision of QoS, such as interstream synchronization, ordered delivery of data, error-recovery mechanism, scheduling mechanism, etc. With the expected services can be connected specific parameters. For example, the interstream synchronization can be defined through an acceptable skew within the particular data stream [7].

The *network QoS parameters* describe requirements on network services. They may be specified in terms of: (1) *network load*, describing the ongoing network traffic and characterized through *average/minimal interarrival* time on the network connection, *packet/cell size*, and *service time* in the node for the connection's packet/cell [18]; (2) *network performance*, describing the requirements which the network services have to guarantee. Performance might be expressed through a *source-to-destination delay* bound for the connection's packet and packet loss rate [18]. Generally, performance bounds are chosen for QoS parameters, such as latency, delay-jitter [2], or bandwidth, but also other parameters for control of QoS (e.g., priority).

[2] Delay jitter is the maximum difference between end-to-end delays experienced by any two packets [15].

2.3.4 QoS Parameter Values and Types of Service

The specification of QoS parameter values determines the types of service. There are distinguished at least three types of service: *guaranteed* , *historical*, and *best-effort* services.

Guaranteed services provide QoS guarantees as it is specified through the QoS parameter values (bounds) either in *deterministic* or *statistical* representation. The deterministic bounds can be given through a *single value* (e.g., average value, contractual value, threshold value, target value), a *pair of values* (e.g., minimal and average value, lowest quality and target quality), or an *interval of values* (lower bound is the minimal value and upper bound is the maximal value). Guaranteed services may deal also with statistical bounds of QoS parameters [18] such as statistical bound on error rate etc.

A *historical service* is based on the past network behavior, hence the QoS parameters are estimates of past behavior which the service tries to match.

Best-effort services are services based either on no guarantees, or on partial guarantees. There is either no specification of QoS parameters required, or some bounds in deterministic or statistical forms are given. Most of the current network protocols have best effort services.

2.4 Resource

A *resource* is a system entity required by tasks for manipulating data. Each resource has a set of distinguishing characteristics [1]:

- There are *active* and *passive* resources. An active resource is, for example, the CPU or a network adapter for protocol processing; it provides a service. A passive resource is, for example, the main memory (buffer space), or bandwidth (link throughput); it denotes some system capabilities required by active resources.

- A resource can be either used *exclusively* by one process at the time or *shared* between various processes. Active resources are often exclusive, passive resources can usually be shared among processes.

- A resource that exists only once in the system is known as a *single resource*, otherwise it is a *multiple resource*. In a transputer based multiprocessor system the individual CPU is a multiple resource.

Services for the multimedia networked applications need resources to perform their functions. Of special interest are resources which are shared among application, system, and network, such

as CPU cycles or network bandwidth. It is important to point out that all shared resources in each layer of MNS can be mapped into three main system resources: *bandwidth of communication channels*, *buffer space*, and *CPU processing power*.

2.5 Relation between Quality of Service and Resources

The QoS parameters specify the resource quantity allocated to the services as well as the service disciplines managing the shared resource in MNS. For example, the end-to-end delay QoS parameter determines the behavior of trausmission services along the path between media source and sink with respect to packet scheduling (bandwidth allocation), queuing (buffer allocation), and task scheduling (CPU processing time allocation).

The above described relation between quality of service and resources is embedded in the form of different mappings between QoS parameters and their corresponding resources in the resource management. Description of a possible realization of resource allocation and management shows the QoS and resource relation. Consider resource allocation and management based on the interaction between clients and the respective resource managers. The *client* requests a resource allocation by specifying its requirements through a QoS specification (this implicitly includes a mapping between the QoS specification and the required resources). This is equivalent to a workload request. The ***resource manager*** checks its own resource utilization and decides if the reservation request can be served or not. All existing reservations are stored, this way their share in terms of the respective resource capacity is guaranteed. Moreover, this component negotiates the reservation request with other resource managers if necessary. Applying the generic scheme on a case, shown in Figure 2, the transmission of video data between a camera, connected to a computer server, and tbe screen of the computer user involves a resource manager for all depicted components.

3 Multimedia Call Set Up Phase

Before any transmission with QoS guarantees in MNS can be performed, several steps have to be executed: (1) The application (or user) defines the required QoS; (2) QoS parameters have to be distributed and negotiated; (3) QoS parameters between different layers have to be translated if their specification is different; (4) QoS parameters have to be mapped to the resource requirements; (5) required resources have to be admitted/reserved/allocated along the path between sender(s) and receiver(s). These steps are performed during the multimedia call set up.

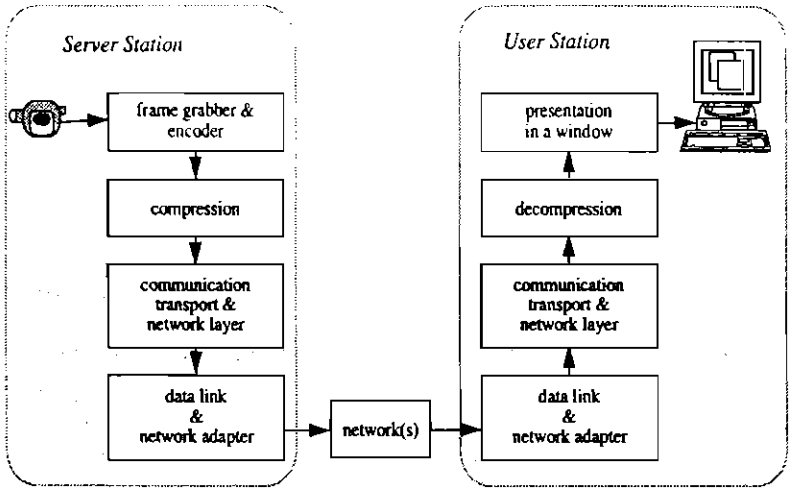


Figure 2: *Components grouped for the Purpose of Video Data Transmission*

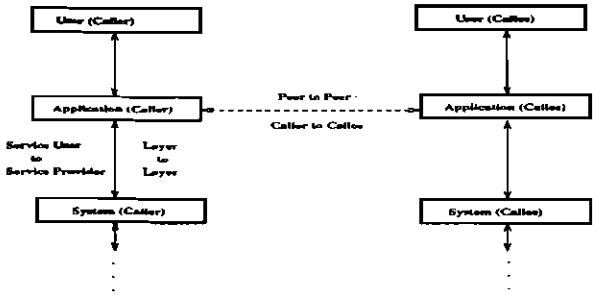


Figure 3: *Negotiation*

3.1 QoS Negotation and Translation

If we assume that the user has defined the networked multimedia application requirements, these requirements have to be communicated to the resource management entities of all involved system components. A general architecture for communication of QoS parameters requires two services: *negotiation of QoS parameters* and *translation of QoS parameters* (if different QoS specification of system components occurs). 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, and *layer-to-layer communication*, which might be, for example, application-to-system negotiation or human-user-to-application negotiation. In ISO terminology the peer-to-peer negotiation is also called caller-to-callee negotiation, and layer-to-layer negotiation is called service-user-to-service-provider negotiation (see Figure 3).

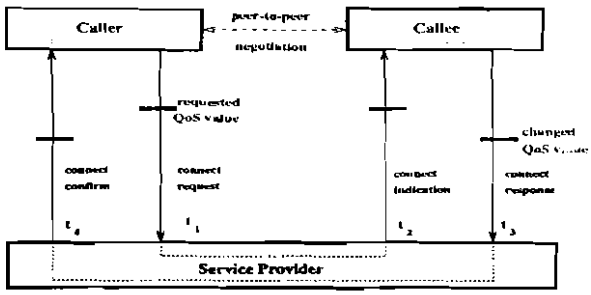


Figure 4: *Bilateral Peer-to-Peer Negotiation*

3.1.1 QoS Negotiation

The purpose of the negotiation is to establish common QoS parameter values among the services users (peers) and service providers (underlying layers). We further assume negotiation of QoS parameters where the QoS parameter values are specified with deterministic bounds (minimal value and average value). There are several possibilities of negotiation among the peers (caller, callee) and the service provider:

- *Bilateral Peer-to-Peer Negotiation*

  This type of negotiation takes place between the two service users (peers), and the service provider is not allowed to modify the value proposed by the service user (Figure 4).

- *Bilateral Layer-To-Layer Negotiation*

  This type of negotiation takes place only between the service user and service provider. This negotiation covers two possible communications: (1) between local service users and providers, for example, between application requirements and OS service guarantees, and (2) between host-sender and the network, for example, when the sender wants to broadcast multimedia streams.

- *Unilateral Negotiation*

  In this negotiation, the service provider as well as the called service user are not allowed to change the QoS proposed by the calling user. This negotiation is reduced to 'take it or leave it' [11]. Further, this negotiation also allows the case in which the receiver may take the proposed QoS and, although may not have the capability to accommodate the QoS parameters, can modify the host-receiver and participate with lower quality on the communication [3].

[3] We see similar case in TV Broadcast. The color TV signal is broadcast uniformly to every user, but users with black and white TV can still watch the TV program, i.e., the control of the quality is done at the receiver device.

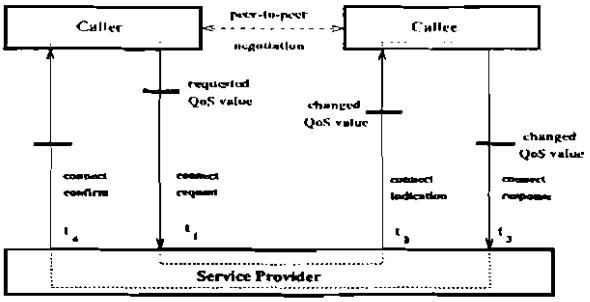


Figure 5: *Triangular Negotiation for Information exchange*

- *Hybrid Negotiation*

  In case of broadcast/multicast communication every participating host-receiver may have different capabilities from host-sender, but still wants to participate in the communication (e.g., conference). Hence, between host-sender and network the QoS parameter values can be negotiated using bilateral layer-to-layer negotiation, and between network and host-receiver can occur unilateral negotiation as described above.

- *Triangular Negotiation for Information Exchange*

  In this type of negotiation, the calling user introduces in the request primitive the average value of a QoS parameter. This value can be changed by the service provider/callee along the path through an indication/response primitive before presenting the final value in the confirm primitive to the caller. At the end of the negotiation, all parties have the same value of QoS parameter (Figure 5).

- *Triangular Negotiation for a Bounded Target*

  This is the same type of negotiation as the previous one, only the values of a QoS parameter are represented through two bounds: target (average value), and the lowest quality acceptable (minimal value). The goal is to negotiate the target value, i.e., the service provider is not allowed to change the value of the lowest quality (if it cannot provide at least the lowest quality, the connection request is immediately rejected) but is free to modify the target value. The callee will make the final decision concerning the selected value of the target. This selected value of the QoS will be returned in confirm primitive to the caller (Figure 6) [11].

- *Triangular Negotiation for a Contractual Value*

  In this case, the QoS parameters are specified through a minimal requested value and bound of strengthening. The goal of this negotiation is to agree on a contractual value, which in

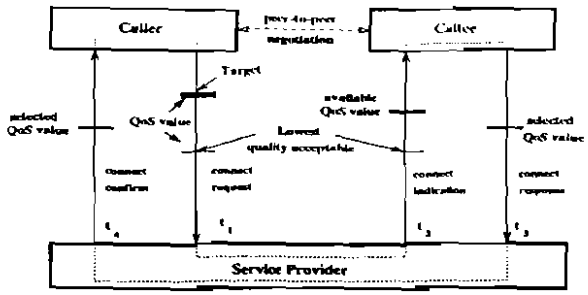


Figure 6: *Triangular Negotiation for Bounded Target*

this case is the minimal request QoS parameter value. The service provider can modify the minimal request value towards the strengthening bound value. The callee makes the final decision and reports with response/confirm primitive to the caller. The contractual value can also be maximal QoS parameter value, or threshold [11] values, which the service user wants to achieve as a contract value.

There are still very few call set-up protocols which have negotiation mechanisms built in. We present some examples which have some notion of negotiation in them: The *Internet Stream Protocol, Version2 (ST-II)* is an IP-layer protocol that provides end-to-end guaranteed service across the Internet network [4]. The parameters related to the throughput are negotiated with a triangular negotiation for a bounded target. For parameters related to delay there is no negotiation. The calling user specifies the maximum transit delay in the connect request. During the establishment of the connection, each ST-agent participating in the stream will have to estimate the average transit delay that it will provide for this stream and the average variance of this delay. The provider presents in the connect indication the total estimated average delay and average variance of delay. The called user decides if the (expected) average delay and delay jitter are sufficient before accepting the connection. The parameters related to the error control are not negotiated.

Other establishment protocols such as *RCAP (Real-Time Channel Administration Protocol* [16], *RSVP* [9] and others use triangular negotiation for different QoS parameter values. The QoS Broker is an end-to-end establishment protocol [2] and includes bilateral negotiation at the application layer between peers, unilateral negotiation with the OS, and triangular negotiation at the transport subsystem layer with underlying ATM network as the service provider (Figure 7).

3.1.2 Translation

It is widely accepted that different MNS components require different QoS parameters; e.g., the mean loss rate, known from packet networks, has no meaning as a QoS for a video capture device

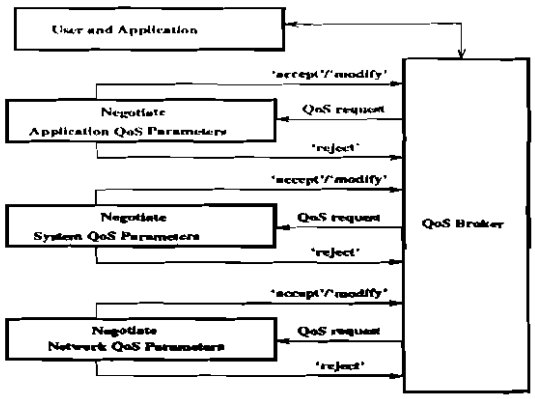


Figure 7: *Negotiation in QoS Broker*

Likewise, the frame quality [4] is of little use to a link layer service provider. We always distinguish between user and application, system and network with different QoS parameters. However, in future systems there may be even more 'layers' or there may be a hierarchy of layers, where some QoS values are inherited and others are specific to certain components. In any case, it must always be possible to derive all QoS values from the user and application QoS values. This derivation –known as *translation*– may require 'additional knowledge' stored together with the specific component. Hence, translation is an additional service for layer-to-layer communication during the call set-up phase. The split of parameters, shown in Figure 1, requires translation functions as follows:

- *human interface – application QoS*

  The service which may implement the translation between a human user and application QoS parameters is called *tuning service.* A tuning service provides a user with an interface for input of application QoS as well as output of the negotiated application QoS. The translation is represented through video and audio clips (in the case of audio-visual media), which will run at the negotiated quality corresponding to, for example, the video frame resolution that end-system and the network can support.

- *application QoS – system QoS*

  Here, the translation has to map the application requirements into the system QoS parameters, which may lead to translation such as from 'high quality' synchronization user requirement to a small (milliseconds) synchronization skew QoS parameter [7], or from video frame size to transport packet size and connected with it possible segmentation/reassembly

[4] Frame quality in terms of number of pixels in both axes is a QoS value to initialize frame capture buffers.

functions.

- *system QoS - network QoS*

  This translation maps the system QoS (e.g., transport packet end-to-end delay) into the underlying network QoS parameters (e.g., in ATM the end-to-end delay of cells) and vice versa.

The important property of the translation service is that it must be *bidirectional translation* [2]. This can cause problems because, for example, the video rate and video frame size together give the throughput parameter for the network. Now, if the throughput bound has to be relaxed, the new throughput value may translate into either lowering the quality of the image or lowering the video frame rate.

At this point, using the previously mentioned 'additional knowledge', a bidirectional translation is possible. In the above mentioned example such a rule may be: (1) reduce the frame size (always keeping the same ratio between horizontal and vertical resolution) until we encounter 112 pixels in the horizontal direction; (2) reduce the frame rate until we have 1 frame per second; and (3) provide an indication that no further reduction is possible and the connection must be closed.

3.2 Resource Reservation

For the provision of guaranteed QoS in MNS, *reservation of resources* is necessary. Without resource reservation and management in end systems and routers/switches, transmission of multimedia data leads to dropped or delayed packets. The reservation of resources is in most systems *simplex*, i.e., the resources are reserved only in one direction on a link, which implies that the senders are logically distinct from receivers. The reservation of resources depends on the *reservation model*, its *protocols* and a set of *resource administration functions* for individual resources. We will discuss in this subsection the reservation model and basic mechanisms of reservation protocols. The resource administration functions, such as allocation of resources, and admission control are discussed in Section 3.3.

3.2.1 Reservation Model

There are three types of reservation model: (1) *Single Sender/Single Receiver* (e.g., RCAP, QoS Broker); (2) *Single Sender/Multiple Receivers* (e.g., ST-II); and (3) *Multiple Senders/Multiple Receivers* (e.g., RSVP).

The reservation model is determined by its *reservation direction and style* [9]. The reservation direction can be *sender-oriented* (e.g., ST-II) or *receiver-oriented* (e.g., RSVP). Sender-oriented reservation means that the sender transmits a QoS specification (e.g., flow specification) to the targets. The intermediate routers and targets may adjust the QoS specification with respect to available resources before the QoS specification is transmitted to the sender. Receiver-oriented reservation means that the receiver describes its resource requirements in a QoS specification and sends it to the sender in a 'reservation' message [9]. It is assumed that a sender has issued a 'path' message before, providing information about outgoing data.

The *reservation style* represents a creation of a path reservation and time when the senders and receivers perform the QoS negotiation and resource reservation. The style for sender-oriented reservation may be either that the sender creates a *single reservation* along the link to the receiver, or the sender creates a *multicast reservation* to several targets. The reservation style for receiver-oriented reservation is defined in RSVP as follows [9]:

- *Wildcard-Filter* style – a receiver creates a single reservation, or resource 'pipe', along each link, shared among all senders for the given session.
- *Fixed Filter* style – each receiver selects the particular sender whose data packets it wants to receive.
- *Dynamic Filter (DF)* – each receiver creates N distinct reservations to carry flows from up to different senders. A later DF reservation from the same receiver may specify the same value of N and the same common flowspec but a different selection of particular senders, without a new admission control check. This is known as *channel switching*, analogous to a television set. If a receiver, using DF reservation style, changes the number of distinct reservations N or the common flow specification, this is treated as a new reservation that is subject to admission control and may fail.

The reservation style can also be divided with respect to the time when the actual resource allocation occurs: (1) *immediate reservation*, and (2) *advanced reservation*. The advanced reservation service is essential in multi-party multimedia applications. There are two possible approaches to the advanced reservation: (1) a *centralized* approach where an advanced reservation server exists, and (2) a *distributed* approach where each node on the channel's path 'remembers' the reservations.

3.2.2 Resource Reservation Protocols

A resource reservation protocol performs no reservation or allocation of required resources itself; it is only a vehicle to transfer information about resource requirements and to negotiate QoS values, which users desire for their end-to-end application. Resource reservation protocols are control protocols embedded in multimedia call set-up protocol. The resource reservation protocol implies that at every node and host exits a *resource manager* which is responsible for sending and receiving the control messages, and invoking the resource administration functions (such as admission control, QoS translation, mapping between QoS and resources, routing, and other management services) needed to make the proper decision for establishing a multimedia call between senders and receivers with QoS guarantees. It means that the resource manager works closely together with network management agents for proper reservation and administration decisions.

The resource reservation protocols work generally as follows: The sender sends QoS specifications in a 'reservation' message (connect request). At each router/switch along the path, the reservation protocol passes a new resource reservation request to the resource manager, which may consist of several components. For example, in RSVP this kind of manager is called a 'traffic controller' and consists of an admission control routine, packet scheduler and packet classifier. After the admission decision, the resource manager reserves the resources and updates the particular service information for QoS provision (e.g., packet scheduler in RSVP).

3.3 Admission/Allocation Service

The tasks of a resource manager cover different phases during the resource admission, allocation, and enforcement process:

- *QoS and Resource Mapping*: The resource manager calculates from the QoS specification the required resource.

- *Resource Admission*: The resource manager checks, using different admission tests, if there is available sufficient capacity of shared resources in order to handle additional resource reservation requests. After the admission test, the resource manager calculates the best possible QoS performance (using the above mentioned mapping QoS-resource function), corresponding to the resources, which can be guaranteed to the new 'reservation' request.

- *Resource Reservation/Allocation*: The resource manager reserves/allocates the required capacity in order to meet the QoS requirements for each request.

- *Resource Scheduling*: Incoming messages on shared resources are scheduled according to the given QoS guarantees during the multimedia transmission. This issue is discussed in detail in Section 4.

In this section we discuss the resource admission and reservation/allocation. The QoS and resource mapping is not discussed here because it is still an open research area and is based on the exact QoS specification of the particular MNS, as well as available resources connected with them, which may be different for various QoS parameter sets and MNS system configurations.

3.3.1 Resource Admission

Admission service is an important service at every node along the path between source (sender) and sink (receiver) to check resources for availability. In the networks, it is the mechanism used to accept or reject new connections; this decision is made during the call set-up. The admission service checks availability of shared resources using availability tests. The resource availability tests are called *admission tests*. Based on the results of the admission tests, reservation protocol creates either 'reserve' message with admitted QoS values [5], or 'reject' message when the minimal bound of QoS values cannot be satisfied.

There are three types of tests which admission should perform: (1) ***schedulability test*** of shared resources such as CPU schedulability, packet schedulability at the entrance to the network and at each network node for delay, jitter, throughput and reliability guarantees; (2) ***spatial test*** for buffer allocation for delay and reliability guarantees; and (3) ***link bandwidth test*** for throughput guarantees.

The admission tests, as mentioned above, depend on the implementation of control (e.g., rate control) mechanisms in the multimedia transmission protocols. There is extensive research into admission control and it is beyond the scope of a survey paper to analyze all existent admission tests, schemes and policies. At this point it is important to emphasize that any QoS negotiation and, therefore, resource admission must be closely related to a *cost function*, for example, *accounting*.

As one example, let us assume we have a *video on demand* service, running in a community. We can save resources if we allow the video clip to be moved to a server 'nearby' the respective client. This can be done more easily if we know prior to the required demand. Hence, a user who 'orders' a video clip for a certain future time (e.g., 1 hour ahead) may pay less than another user who chooses some video clip and immediately wants to access it. If the client is not forced to pay,

[5] The admitted QoS values may be lower than the target value, but they may be still above the minimal value as it is shown in Figure 5 and 6.

he/she will always demand the best available QoS. In this case, some other clients may end up with a reduction in quality or not using this service at all because this is the only result they get through any QoS negotiation. With the introduction of appropriate accounting, QoS negotiation may well become a real negotiation.

3.3.2 Resource Reservation/Allocation

Reservation/allocation of resources can be made either in a *pessimistic* or in an *optimistic* way:

- The *pessimistic approach* avoids resource conflicts by making reservations for the worst case, e.g., reservation for the longest processing time of the CPU or the highest bandwidth needed by a task is made. Resource conflicts are therefore avoided. This leads potentially to an underutilization of resources. This method results in a guaranteed quality of service.

- The *optimistic approach* reserves resources according to an average workload. In the case of the above mentioned example, CPU is only allocated for the average processing time. This approach may overload resources when unpredictable behavior occurs. QoS parameters are met as far as possible. Resources are highly utilized, though an overload situation may result in failure. A *monitor function* to detect the overload and to solve the problem should be implemented. The monitor function then preempts processes according to their importance.

The optimistic approach is considered to be an extension of the pessimistic approach. Additional mechanisms to detect and solve resource conflicts have to be implemented. The resource managers (e.g., in HeiRAT [4]) may provide the following data structures and functions for resource reservation:

- *Resource Table*: A resource table contains information about the managed resources. This includes static information like the total resource capacity available, the maximum allowable message size, the scheduling algorithm used, dynamic information like pointers to the connections currently using the resource, and the total capacity currently reserved.

- *Reservation Table*: A reservation table provides information about the connections for which portions of the managed resources are currently reserved. This information includes the QoS guarantees given to the connections and the fractions of resource capacities reserved for these connections.

- *Reservation Function*: A reservation function, used during the call set-up phase, calculates the QoS guarantees that can be given to the new connection and reserves the corresponding resource capacities.

4 Multimedia Data Transmission Phase

QoS guarantees must be met in the application, system and network (Figure 1) in order to get the acceptance of the users of MNS. There are several constraints which have to be satisfied in order to provide guarantees during multimedia transmission: (1) *time constraints* which include delays; (2) *space constraints* such as system buffers; (3) *device constraints* such as frame grabbers allocation; (4) *frequency constraints* which include network bandwidth and system bandwidth for data transmission; and (5) *reliability constraints*. These constraints can be specified if proper resource management is available at the end-points as well as in the network. However, these five constraints are related to each other in such a way that one parameter may imply choosing another. For example, time constraints for the scheduling of video frame data imply a corresponding bandwidth allocation.

We discuss in the following subsection process management of the operating system to provide timing constraints for multimedia task scheduling, buffer management for provision of space constraints, rate control mechanisms for delay, delay-jitter and throughput (bandwidth) provision as well as error control for reliability provision. We assume at this point that proper resource reservation and allocation has occurred as described in the previous sections.

4.1 Process Management

The process management deals with the resource *processor*. The capacity of this resource is specified as *processor capacity*. The process manager maps processes onto the resource 'processor' according to a specified scheduling policy so that all processes meet their requirements. The process manager is called the *scheduler*. This component manages the single process state. The *dispatcher* manages the transition from the state *ready to run* to *run*. In most traditional operating systems, the next process to run is chosen according to a priority policy. Between processes with the same priority the one with the longest *ready time* is chosen.

Continuous media data require processing at the application level as well as at the system level in exactly predetermined, usually periodic, intervals. The computing of this data has to be completed by certain deadlines; hence we need a real-time process scheduler, which means that the manager has to determine a schedule for the resource CPU that gives processing guarantees.

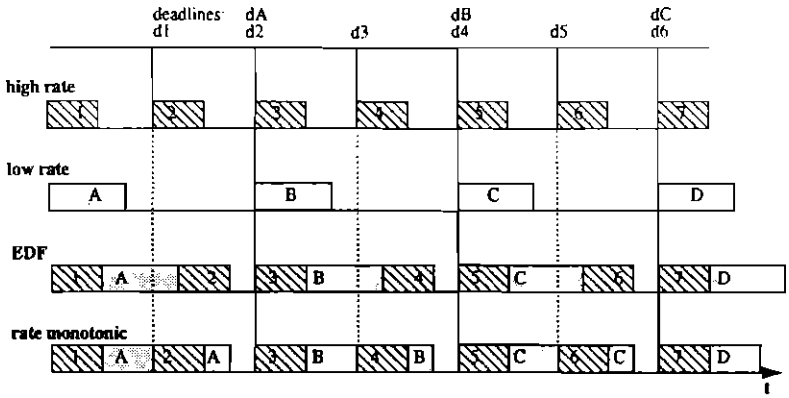


Figure 8: *Rate Monotonic versus EDF: Context Switches in Preemptive Systems*

This must be guaranteed for all tasks [6] in every period for the whole run time of the system. In a multimedia system, continuous and discrete media are processed concurrently.

These real-time processing requirements of multimedia tasks result in two conflicting goals [1]:

- A non-real-time process should not suffer from starvation because real-time processes are executed. Multimedia applications rely on text and graphics as much as on audio and video. Therefore, not all resources should be occupied by the real-time processes and their management processes.

- A real-time process must never be subject to *priority inversion*. The scheduler has to ensure that any priority inversion (also between time-critical processes with different priorities) is avoided or reduced as far as possible.

The cost of the scheduling of every message should be minimized. The continuous media data are more critical because they occur periodically during the processing of real-time tasks. The overhead (context switching) generated by the scheduling and operating system is part of the processing time and therefore has to be added to the processing time of the real-time task.

We describe some scheduling algorithms which are suitable for scheduling multimedia tasks, as shown in Figure 8, and implemented in some multimedia operating systems [1]:

- *Earliest Deadline First Algorithm (EDF)*

  At every new ready state, the scheduler selects among tasks that are (1) ready and (2) not fully processed but with the earliest deadline. The requested resource is assigned to the selected task. At any arrival of a new task, EDF immediately computes its deadline and preempts

[6] In our case, only periodic tasks without precedence constraints are discussed, i.e., the processing of two tasks is mutually independent.

the running task if the new task deadline is earlier than that running. The processing of the interrupted task is continued later according to EDF. EDF is an optimal, dynamic algorithm, which means that it produces a valid schedule whenever one exists. EDF can also be extended to be used with non-preemptive systems. With a priority driven system scheduler, each task is assigned a priority according to its deadline. The highest priority is assigned to the task with the earliest deadline, the lowest to the one with the latest. With every new task, the priorities might have to be adjusted. When applying EDF to the scheduling of continuous media on a single processor machine with priority scheduling, process priorities are likely to be rearranged quite often which may cause considerable overhead.

- *Rate Monotonic Algorithm*

  This algorithm is an optimal, static, priority-driven algorithm for preemptive, periodic tasks. A process is scheduled by a static algorithm at the beginning of the processing. Subsequently, each task is processed with the priority calculated at the beginning. No further scheduling is required. The following five assumptions are necessary prerequisites to apply the rate-monotonic algorithm. (1) The requests for all tasks with deadlines are periodic, i.e., with constant intervals between consecutive requests. (2) The processing of a single task has to be finished before the next task of the same data stream becomes ready for execution. Deadlines consist of run-ability constraints only, i.e., each task must be completed before the next request occurs. (3) The request of tasks is independent, i.e., the requests for a certain task do not depend on the initiation or completion of requests for any other task. (4) Run-time for each request of a task is constant. Run-time denotes the maximum time which is required by a processor to execute the task without interruption. (5) Any non-periodic task in the system has no required deadline.

  Further work has shown that not all of these assumptions are mandatory to employ the rate-monotonic algorithm [14]. Static priorities are assigned to tasks once during call set-up phase according to their request rates. The priority corresponds to the importance of a task relatively to other tasks. Tasks with higher request rates will have higher priorities. The task with the shortest period gets the highest priority and the task with the longest period the lowest priority.

As described above, the scheduling for multimedia may be implemented through priority mechanism because this is the algorithm which the current operating systems, with their real-time ex-

tension, offer (e.g., Real-Time Extension of AIX, IRIS's real-time extension REACT). Therefore, we will briefly discuss two important issues of priority implementation for MNS:

- *Priority Inversion Problem*

  A real-time designer needs to determine the worst case blocking time of a higher priority activity accessing a shared resource protected by a critical region. It is often impossible to compute the bound if an activity in the protected region is preemptable. For implementing predictable computing, blocking time should be bounded. The unbounded blocking is caused when lower priority activities preempt higher priority activities. This is called the ***priority inversion problem*** [17].

- *Priority Inheritance*

  In order to bound the worst case blocking time, the ***priority inheritance*** scheme was developed [17], meaning that all tasks processing a certain priority message should propagate their priority to other tasks processing this priority message. For example, if we have a client-server system, and the client sends messages with high priority, which means that tasks processing these messages are also scheduled at high priority, then the server should also process the message at the same high priority. This implies that the priority, which is assigned by the application, is propagated to the network protocol processing, as well as to the remote sides. This propagation of the priority information can be performed during the QoS negotiation process.

In traditional operating systems, FCFS (First Come First Serve) policy is adopted for Inter-process Communication (IPC) message handling, since fairness is important for providing virtually infinite resources and avoiding starvation. In real-time systems, however, FCFS ordering often creates the priority inversion problem since a higher priority activity must wait for completion of all low priority activities in a waiting queue. Thus, MNS should provide a priority based ordering for queuing requests to avoid the priority inversion problem. This also implies that the scheduling of the shared resources for MNS has to be synchronized with respect to priorities. Otherwise, the delay bounds are unpredictable and cannot be guaranteed.

4.2 Buffer Management

A severe problem is the host systems' limited memory bandwidth. To enable high-speed multimedia distributed applications, a tighter integration of the data flow through the communication system

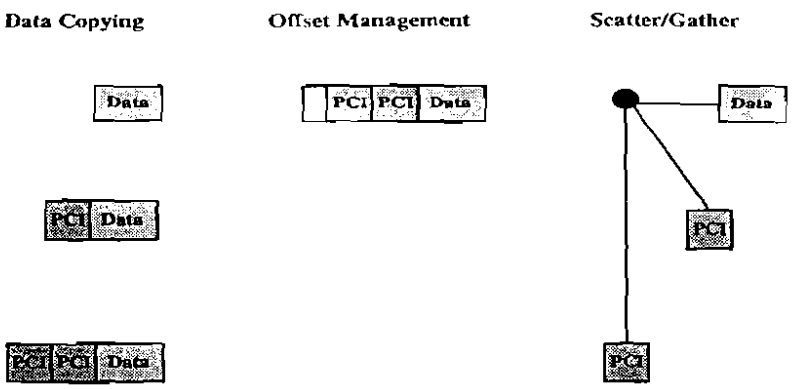


Figure 9: *Different Buffer Management Techniques*

and application is needed. We describe several mechanisms for buffer management (Figure 9) and evaluate them for MNS:

- *Data Copying*

  A buffer management technique based on *data copying* across layer boundaries allows perfect separation of different layers. Each layer can allocate exactly the amount of memory needed to hold the packet which has been currently processed. However, the protocol processing normally needs much less memory and CPU bandwidth than the movement of data, which is done twice in the layer for receiving and transmitting again. This technique is expensive and not suitable for MNS because the protocol data unit (PDU) sizes can become quite large (e.g., video).

- *Offset Management*

  In an *offset management* approach, a buffer is placed in one memory segment and the buffer must be large enough to contain all data to be transmitted, as well as the protocol control information (PCI). For transmission, an application will place its data into the buffer and leave enough space for the future PCIs of the protocol stack. As the buffer is passed through the communication system, each protocol entity updates the offset in the buffer which means it writes in the same buffer its own PCI. The offset technique is easy to implement with little processing overhead for each buffer and prevents data copying. The problem is that the application has to allocate the maximum sized buffer to ensure that there is enough space for all protocol information. This implies that the application needs to know the sizes of all PCIs in case no segmentation happens. This also means that this technique can be used between layers where the data unit is not segmented/reassembled.

- *Scatter/Gather System*

  In a scatter/gather system each request for space in a buffer is satisfied by linking in a new memory segment. A control structure is used to keep track of all the memory segments for a buffer. The scatter/gather system prevents data copying, and also allocates memory space only as it is needed, not causing memory over-allocation. This technique, used in the χ-kernel, enables the efficient implementation of protocol requirements such as dynamic expansion of buffers, segmentation and reassembly, and concatenation and separation. A scatter/gather buffer management system needs only to mark the memory segments as having multiple references to them, and to create more control structures referencing the same memory segments. However, additional overhead for the control structures are incurred. This has prompted modifications to the χ-kernel's scatter/gather buffer management, such as support of an additional stack structure for PCI and not using individual memory segments for each PCI.

We foresee the solution for a good buffer management in MNS as a combination of the scatter/gather technique with offset management in the individual memory segments. The implementation of these buffer management techniques, using different OS support for transferring data from one domain to another, (e.g., shared memory [6], page remapping, fbufs) is an important issue which needs to be point out, although it is beyond the scope of this paper.

4.3 Rate Control

If we assume a MNS to be a tightly coupled system, which has a central instance managing all system components, then this central instance can impose a synchronous data handling over all resources; we encounter a fixed, imposed data rate. However, a MNS usually comprises loosely coupled end-systems which communicate over networks. In such a set-up, rates have to be imposed. Here, we make use of all available strategies in the communications environment.

High speed networking provides opportunities for multimedia applications to have stringent performance requirements in terms of throughput, delay, delay-jitter and loss rate. Conventional packet switching data networks with window-based flow control and FCFS cannot provide services with strict performance guarantees. Hence, for MNS new *rate-based flow control* and *rate-based service disciplines* are being introduced. These control mechanisms are connected with a connection-oriented network architecture which support explicit resource allocation and admission control policies.

A *rate-based service discipline* is one that provides a client with a minimum service rate inde-

pendent of the traffic characteristics of other clients. Such a discipline, operating at a switch [7], manages the following resources: *bandwidth, service time (priority)*, and *buffer space*. Together with proper admission policies, such disciplines provide throughput, delay, delay jitter and loss rate guarantees. Several rate-based scheduling disciplines have been developed [15]:

- *Fair Queuing*

  If N channels share an output trunk then each one should get *1/Nth* bandwidth. If any channel uses less bandwidth than its share, then this portion is shared among the rest equally. This mechanism can be achieved by the ***bit-by-bit round robin (BR)*** service among the channels. The BR discipline serves n queues in the round robin service, sending one bit from each queue that has a packet in it. Clearly, this scheme is not efficient; hence Fair Queuing emulates BR as follows: each packet is given a finish number, which is the round number at which the packet would have received service, if the server had been doing BR. The packets are served in order of the finish number. Channels can be given different fractions of the bandwidth by assigning them weights, where weight corresponds to the number of bits of service the channel receives per round of BR service.

- *Virtual Clock*

  This discipline emulates *Time Division Multiplexing (TDM)*. To each packet a virtual transmission time is allocated. It is the time at which the packet would have been transmitted, if the server would actually be doing TDM.

- *Delay Earliest-Due-Date (Delay EDD)*

  The Delay EDD [18] is an extension of EDF scheduling (Earliest Deadline First) where the server negotiates a service contract with each source. The contract states that if a source obeys a peak and average sending rate, then the server provides bounded delay. The key then lies in the assignment of deadlines to packets. The server sets a packet's deadline to the time at which it should be sent, if it had been received according to the contract. This actually is the expected arrival time added to the delay bound at the server. By reserving bandwidth at the peak rate, Delay-EDD can assure each channel a guaranteed delay bound.

- *Jitter Earliest-Due-Date (Jitter-EDD)*

  Jitter EDD extends Delay-EDD to provide delay-jitter bounds. After a packet has been served at each server, it is stamped with the difference between its deadline and actual finishing time.

[7] The term 'switch' is used in ATM networks, where 'router' or 'gateway' are used in Internet environment.

A regulator at the entrance of the next switch holds the packet for this period before it is made eligible to be scheduled. This provides the minimum and maximum delay guarantees.

- *Stop-and-Go*

  This discipline preserves the 'smoothness' property of the traffic as it traverses through the network as follows: the main idea is to treat all traffic as frames of length T bits, meaning, the time is divided into frames. At each frame time, only packets that have arrived at the server in the previous frame time are sent. It can be shown that the delay and delay-jitter are bounded, although the jitter bound does not come free. The reason is that under Stop-and-Go rules, packets arriving at the start of an incoming frame must be held by full time T before being forwarded. So all the packets that would arrive quickly are instead being delayed. Further, since the delay and delay-jitter bounds are linked to the length of the frame time, improvement of the Stop-and-Go can be achieved using multiple frame sizes, which means it may operate with various frame sizes.

- *Hierarchical Round-Robin (HRR)*

  HRR server has several service levels where each level provides round-robin service to a fixed number of slots. Some number of slots at a selected level are allocated to a channel and the server cycles through the slots at each level. The time a server takes to service all the slots at a level is called the *frame time* at the level. The key of HRR is that it gives each level a constant share of the bandwidth. 'Higher' levels get more bandwidth than 'lower' levels, so the frame time at a higher level is smaller than the frame time at a lower level. Since a server always completes one round through its slots once every frame time, it can provide a maximum delay bound to the channels allocated to that level.

Rate-based service disciplines need to allocate resources per client, hence the clients need to specify their traffic (using QoS parameters). The traffic specification for Virtual Clock, HRR and Stop-and-Go are : a *transmission rate* averaged over an *interval.* Delay-EDD and Jitter-EDD have three parameters: *minimal packet inter-arrival time*, *average packet inter-arrival time*, and *interval* over which the average packet inter-arrival time was computed. Fair Queuing was described for datagram networks, so no traffic specification was proposed. Rate-based disciplines can be further classified depending on the policy they adopt: (1) the *Work-conserving discipline* serves packets at the higher rate as long as it does not affect the performance guarantees of other channels which also means a server is never idle when there is a packet to be sent (Delay-EDD, Virtual Clock,

Fair Queuing); and the (2) *Non-work-conserving discipline* does not serve packets at a higher rate under any circumstances which also means that each packet is assigned, explicitly or implicitly, an *eligibility time*. Even when the server is idle, if no packets are eligible, none will be transmitted (Stop-and-Go, HRR, Jitter-EDD).

It was shown in [15] that the buffer space requirements for the three non-work-conserving disciplines are almost constant for each node traversed by the channel. The buffer space requirement for work-conserving delay-EDD increases linearly for each node along the path. Throughput guarantees are provided by all rate-based services. Delay guarantees are provided only by Delay-EDD and all non-work-conserving services. Jitter guarantees are provided by Stop-and-Go and Jitter-EDD.

4.4 End-to-End Error Control

Multimedia extensions to existing operating systems provide a fast and efficient data transport between sources and destination located at the same computer. Glitches on video streams may (but should not) occur, but audio is always conveyed in a reliable way. The solution becomes different if we take into account networks. In the past, several multimedia communication systems have been proposed which usually offer unreliable transport. For example, UDP/IP protocol was used for experiments to transmit digital audio over the Internet. Other examples are the Tenet protocol suite's transport protocols RMPT (Real-Time Message Transport Protocol) and CMTP (Continuous Media Transport Protocol) which provide unreliable but timely delivery for multimedia communication.

A substantive degree of reliability in MNSs is necessary because of following: (1) *Decompression Technology.* Most audio and video compression schemes cannot tolerate loss; they are unable to resynchronize themselves after a packet loss or at least visible or/and other perceptual errors are introduced. (2) *Human Perception.* Loss of digital audio, for example, is detected by a human ear very quickly and results in lower acceptance of the multimedia system. (3) *Data Integrity.* For example, in a recording application one cannot recover from an error that is induced in the first recording of data. Fortunately, in this type of application, where multimedia data is written to disc, there are often less stringent real-time requirements for the receiver.

4.4.1 Error Detection

Reliability should be enforced although there is some error tolerance in the multimedia systems. This works, however, only if the *application* is able to isolate the errors. For example, some wrong colors within a video frame may not matter because they are hardly visible to the human user as

they appear for a short fraction of a second, but if the frame boundaries are destroyed there is no way to recover from the error. This means that structural information within a data stream needs to be protected, content not always. This also implies that existing error detection mechanisms such as *checksumming* and *PDU sequencing* have to be extended towards conveying further information. These existing mechanisms allow detection of data corruption, loss, duplication, and misorder at the lower levels (e.g., packets in the transport layer), but on the application PDU level, where actually the decision should be made, if the packet is lost or not, error detection is left out.

Another example for enforcing error detection at a higher layer (above the transport layer) is MPEG-2 encoded video. This compression produces three type of frames in the video streams. The most important frame type is the I-frame which contains the structural information of the video stream for a certain time interval. The two other types of video frames (P-frame and B-frame) follow the I-frame with supporting information. Hence, it is important for the multimedia communication system not to lose the I-frame (strict reliability requirements on the sequence of I-frames), but there is a certain tolerance towards losses of P-frames or B-frames.

In the transport and lower layers the error detection mechanisms have to be extended too because of the 'lateness' concept. It is a new error. The mechanism to detect late data goes as follows. To identify late data it is necessary to determine the lifetime of PDUs and compare their actual arrival time with their latest-expected arrival time. The latest-expected arrival time can be derived from the traffic model (throughput and rate) associated with a connection. This means that for continuous streams the expiration time can be calculated from the PDU rate. Therefore, only the first PDU has to carry a time stamp, although this is not an ideal solution because error detection is forced to start with the first PDU, and no interruption of the service is possible. With a time stamp in every PDU the error detection can start at any point during the media transmission. This mechanism requires a synchronized system clock at the sender and receiver to allow an accurate determination of the end-to-end delay. A possible protocol for this kind of synchronization is Mill's *Network Time Protocol (NTP)*.

4.4.2 Error Correction

The traditional mechanism for reliability provision is *retransmission* (e.g., TCP protocol uses this mechanism) which uses either acknowledgment principle after receiving data or window-based flow control. If the acknowledgment is negative, the data is re-sent by the sender. The traditional reliable transfer strategies are not suitable for multimedia communication because: (1) with explicit

acknowledgment the amount of data to be stored at the sender for potential retransmission can become very large (e.g., in the case of video); (2) with the traditional window-based flow control, the sender may be forced to suspend transmission while a continuous data flow is required; (3) the retransmitted data might be received 'too late' to be consumed in real-time; (4) traditional mechanisms also do not scale to multiple-target communication – they are not designed for multicasting communication only for point-to-point communication. We will outline some error correction schemes for MNS, currently discussed in research:

- *Go-back-N Retransmission*

  This method is the most rigid error correction scheme. The mechanism is as follows: if PDU i is lost, the sender will go back to i, and restart transmission from i. The successive PDUs after i are dropped at the receiver. The lost PDU is recovered only if $i \leq n$ where n is specified at the beginning of the transmission. This means it is specified (n) how far back the data should be retransmitted if a packet is lost. This is a simple protocol where no buffering or resequencing of the PDUs at the receiver are necessary. The receiver only send a negative acknowledgment if PDU i is lost. The problem is that if after i's PDU the packets were transmitted successfully, they are dropped too, which may lead to several implications: (1) ***gap introduction*** (Figure 10); and (2) ***violation of throughput guarantees.*** The retransmission introduces gaps because the receiver has to wait at least $2 \times end - to - end\ delay$ to get the proper PDU i. Also, for a multimedia connection where throughput guarantees are provided through rate control, the retransmitted data have to be sent either 'on top' of the guaranteed throughput, or the retransmitted PDU will fall under the rate control. This again leads to a gap in the stream presentation which needs to be handled properly through mechanism such as freeze the video, or turn down audio.

- *Selective Retransmission*

  Selective retransmission provides better channel utilization. The receiver sends negative acknowledgement to the sender if PDU $i \leq n$ is lost. The sender retransmits only those PDUs which have been reported missing, not the consecutive packets too. The disadvantage of this mechanism is its complicated implementation. At the receiver, every successfully received PDU has to be stored until all previous PDUs have been received correctly. It has been shown that this resequencing is worthwhile only if the receiver is able to store at least two times the data corresponding to the bandwidth-delay product.

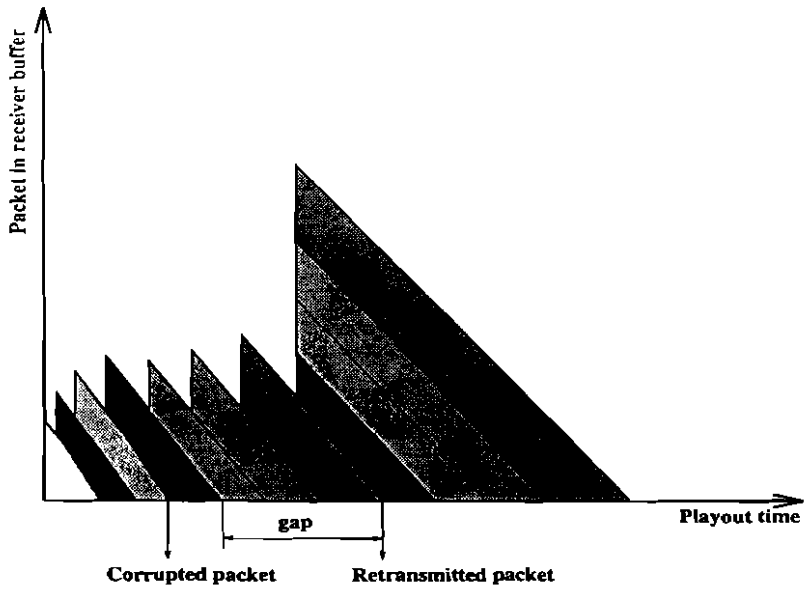


Figure 10: *Gaps in Go-back-N Retransmission*

- *Partially Reliable Streams*

  Partially reliable streams introduce a weak concept of reliability. This mechanism limits the number of packets to be retransmitted. Only the last n packets of the stream in a certain time interval will be retransmitted. The value n can be calculated from the timing constraint of the multimedia application, taking into account the reliability of the underlying network. The possible n can be negotiated during the call set-up between the sender and the receiver.

- *Forward Error Correction (FEC)*

  In this mechanism the sender adds additional information to the original data such that the receiver can locate and correct bits or bit sequences. FEC for ATM networks is discussed in [12]. A given FEC mechanism can be specified by its code rate C (code efficiency), which can be computed: $C = \frac{S}{(S+E)}$; S represents the number of bits to be sent, E represents the number of added check bits. The redundancy introduced by the mechanism is $(1 - C)$ and it must be determined by the transport system. The transport system needs two informations: (1) the error probability of the network between the sender and receiver; and (2) reliability, required from the application. FEC results in a low end-to-end delay and there is no need for exclusive buffering of data before play-out. It also does not require a control channel from the receiver to the sender. The disadvantage of FEC is that it works only for error detection and correction within a packet but not for complete packet loss, i.e., FEC cannot guarantee that corrupted or lost packets can always be recovered. Further, FEC increases the demand

on throughput significantly. The negative effects of added congestion on a network due to FEC overhead can more than offset the benefits of FEC recovery [12]. Also, FEC requires hardware support at end-systems in order to encode and decode the redundant information with sufficient speed. FEC is also used for storing audio data at Compact Disc (CD) devices.

- *Priority Channel Coding*

  Priority channel coding refers to a class of approaches that separates the medium (e.g., voice) into multiple data streams with different priorities. These priorities are then used to tag voice packets so that during periods of congestion the network is more likely to discard low priority packets which carry information less important for reconstructing the original media stream. This scheme enables multiple-priority channels to maintain a higher QoS over larger loss ranges than channels using a single priority for all voice packets. Channel coding requires that the network be able to control packet loss during congestion through a priority mechanism. The use of different streams for different priorities requires synchronization at a per-packet granularity in order to reconstruct the voice signal. Another example where prioritized transmission can be used is for MPEG-2 encoded video. Here, I and P frames could be sent at high priority and B frames could be sent at low priority. Network switches drop lower priority cells or provide a lower grade of service during periods of network congestion.

- *Slack Automatic Repeat Request (S-ARQ)*

  S-ARQ is an error control scheme based on retransmission of lost voice packets in high-speed LANs. The packets are subject to delay-jitter, hence the receiver observers *gaps*, which result in interruptions of continuous playback of the voice stream. Delay-jitter in packetized voice transmission is commonly addressed through a control time at the receiver. The first packet is artificially delayed at the receiver for the period of the control time in order to buffer sufficient packets to provide for continuous playback in the presence of jitter. The voice data consist of talkspurts and periods of silence. Since talkspurts are generally isolated from each other by relatively long silence periods, voice protocols typically impose the control time on the first packet of each talkspurt. The *slack time* of a packet is defined as the difference between its arrival time at the receiver and its playback time, which is the point in time at which playback of the packet must begin at the receiver in order to achieve a zero-gap playback schedule for the talkspurt. Due to delay-jitter, a packet may arrive before or after its playback time. In the former case, the packet is placed in a *packet voice receiver* queue until it is due for

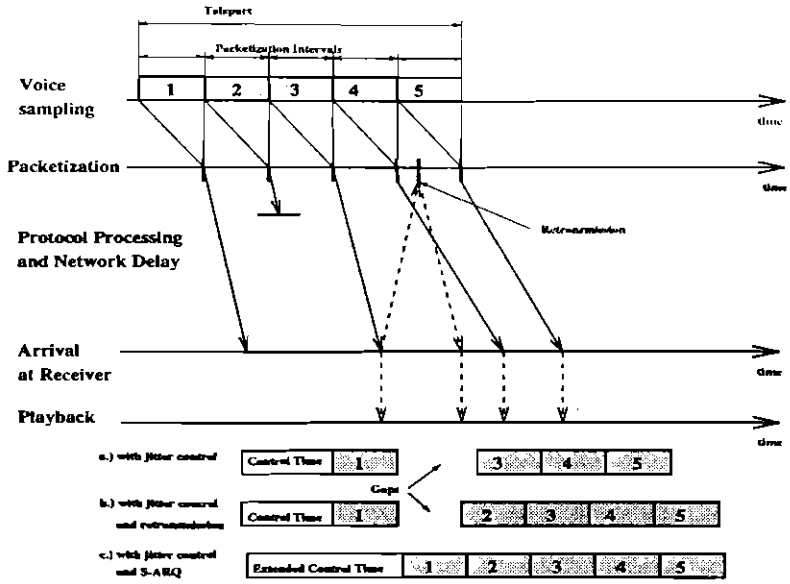


Figure 11: *S-ARQ*

playback. In the later case, a gap has occurred and the packet is played immediately. The principle of S-ARQ is to extend the control time at the beginning of a talkspurt and use it so that the slack time of arriving packets is lengthened. An example is shown in Figure 11 [8].

The error control/correction schemes for multimedia communication systems, as described above, can be divided into two classes: (1) *partial retransmission mechanisms* (e.g., Go-Back-N, Selective retransmission, Partially Reliable streams, S-ARQ); and (2) *preventive mechanisms* (e.g., FEC, Priority Channel Coding). All partial retransmission schemes lack the possibility of introducing a discontinuity or working properly if we introduce large end-to-end delays with large buffers. Hence, preventive schemes should be used.

4.5 Resource Administration

Resource administration includes functionalities such as set-up of resources (allocate), tear down resources (deallocate) and monitor resources (status inquiry).

The resources are allocates during the multimedia call set-up. Section 3 described the services such as negotiation, reservation, admission, and allocation of resource which participate in call (channel, connection) set-up with QoS guarantees. In this section we will concentrate on resource deallocation, resource monitoring and resource administration protocols.

4.5.1 Resource Deallocation

After the transmission of media, resources have to be deallocated, which means, the CPU, network bandwidth, buffer space must be freed and the connections through which the media flow have to be torn down. The tear down process has to be done without disruption of other flows in the network. Further, the tear down process implies updating the resource availability by the resource manager. The resource deallocation is divided with respect to the direction of the request. (1) *Sender* requests closing of the multimedia call. This implies that the resources for all connections corresponding to the multimedia call along the path between sender and receiver(s) have to be deallocated and the resource availability has to be updated at every node; (2) *Receiver* requests closing of the multimedia call. This request is sent to the sender and during the traversing the path the resources are deallocated.

The deallocation depends on the mechanisms of resource management at the nodes and on the tear down protocols.

4.5.2 Resource Monitoring

Resource monitoring is an important part of resource management in networks as well as at end-points. Resource monitoring functionality is embedded in the resource manager and is closely connected to the network management agent[8], i.e., it can use the Management Information Base (MIB) for resource admission. In order to use network management for resource management, the MIBs of the network management are to be extended for multimedia communication with the QoS parameters. Further, network management may be enhanced with functionalities for QoS supervision and problem resolving functions.

Monitoring in networks can add overhead during multimedia transmission, which should not cause violation of QoS guarantees. Hence, monitoring should be flexible, which means that (1) most of the monitoring variables should be optional; and (2) monitoring should be able to be turned on and off [4]. There are two possible modes to operate resource monitoring: *end-user mode* and *Network mode*. The former requests a status report about the resources; the later reports regularly the resource status on different nodes along the path between the communicating end-users.

[8] Network management works with *agents* at every intermediate network node (routers/switches), which are concerned with (1) the information which can be exchanged among the intermediate network nodes, and the structure of this information, stored in the *Management Information Base* (MIB); and (2) the protocol used to exchange information between the network agent and the managed component (e.g., resource manager).

Monitoring at end-systems includes a *supervisor function* to continuously observe if the processed QoS parameters do not exceed their negotiated values. As an example, a compression component may allow delivery at a peak rate of 6 Mbit/s over a duration at most of 3 frame lengths. However, at some point in time the system starts to deliver a continuous bit rate of 6 Mbit/s. The monitoring function will detect this behavior by being called from an exception handler of the rate control component: a buffer overflow occurred at the sender – something which should never happen. The monitoring function finds out that the origin of the exceeded QoS value is an erroneous compressing component. It should be pointed out that the design and implementation of such a monitoring function is a non-trivial task and that a clearly defined notion of the QoS is a prerequisite.

4.5.3 Resource Administration Protocols

Resource Administration Protocols provide communication about resources between individual resource managers at the intermediate nodes and end-points during multimedia transmission. They can be implemented either as part of the *network management protocols* or as separate *resource management protocols.* If the former, the resource administration protocols may be embedded in the following currently standard network management protocols: (1) *CMIS/CMIP (Common Management Information Services and Protocol)*, OSI family standards which are applied in the wide area network environments; and (2) *SNMP (Simple Network Management Protocol)*, a protocol currently prevalent in Local Area Networks. SNMP is based on the Internet Protocol. An example of a separate resource administration protocol is RCAP in Tenet protocol stack.

The administration protocols may be connected with their transmission protocols but do not have to be. An example, where administration protocol is decoupled from transmission protocols, is RCAP [16] RCAP is decoupled from the RTIP (Real-Time Internet Protocol), RMTP (Real-Time Message Transport Protocol) and CMTP (Continuous Media Transport Protocol). The user can request status from RCAP about the resources, but the transport/network protocols do not interact during the transmission with RCAP. An example for coupling the establishment, transmission and management phase is ST-II protocol. ST-II is further coupled to SNMP-MIB for management.

5 QoS Renegotiation and Adaptive Schemes

In continuous media communication, it is important to support a framework capable of dynamically changing the network capacity of each session. Hence, it is important for the MNS architecture

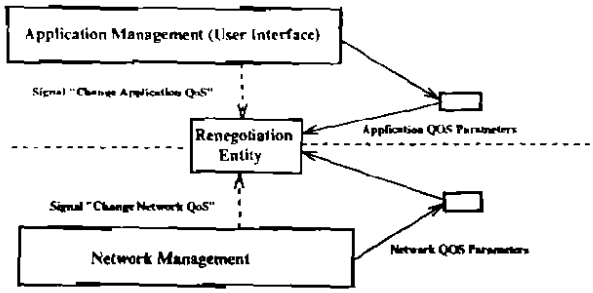


Figure 12: *Signaling Paradigm for Renegotiation at the End-Point*

to support dynamic change of QoS parameters, so that they can be balanced to reach an optimal value for all sessions in a predictable manner.

There are two important factors which have to be provided in order to achieve this goal: (1) notification and renegotiation of QoS parameters , i.e., a protocol for reporting the QoS changes and modifying QoS parameters of existing connections (may be done by resource administration protocols); and (2) adaptive resource schemes to respond to and accommodate the changes either in the network, the hosts, or both.

5.1 Renegotiation of QoS Parameters

During the multimedia transmission, change of QoS parameters and with them associated resources can occur. If such changes occur, *renegotiation of QoS parameters* must begin. Hence, renegotiation is a process of QoS negotiation when a call is already set-up. The renegotiation request can come either from the user, who wants to change the quality of services, or from the host system due to overload of the workstation (multi-user, multi-process environment) or from the network due to overload and congestion. The renegotiation request is sent to a *renegotiation entity* which may be the resource manager itself, or the network agent, or an other separate entity. The implementation depends on the MNS designer. The renegotiation entity processes the request, either by itself (in this case the renegotiation entity is the resource manager) or delegating the request to the resource manager (in this case the renegotiation entity is the network agent or some other entity). A possible signaling paradigm for renegotiation at the end-point is shown in Figure 12.

5.1.1 User Request for Renegotiation

If the user-sender requires a change of QoS, this may imply adaptation of multimedia sources and local host resources as well as network resources. The renegotiation entity has to check if local resources are available. We assume that the renegotiation request can be accommodated at the

multimedia source, meaning, that, for example, the user does not require 30 frames/second if the video encoder can provide only 10 frames/second. Further, the resource administration protocol has to be invoked to check (using resource managers) the availability of network resources if the change of QoS requires change of network resources. If resources are available, then resource reservation and allocation is performed.

If the user-receiver requires change of QoS for the receiving media, first the renegotiation entity checks the local resource and reserves it. Then the sender is notified via resource administration protocol, and the same admission procedure follows as in the case of a user-sender requiring QoS changes. At the end, the receiver has to be notified to change the local resource allocation. In a broadcast or multicast communication structure, different QoS values may be applied for the same connection to different end-systems.

5.1.2 Host System Request for Renegotiation/Change

This request may come from the operating system, if such a capability is provided, in a multi-user environment. In this case, several users are admitted and some of the users (misbehaved user) violate their admitted application requirements. Then a notification about the degradation of the QoS performance and renegotiation request occur. The response is either adaptation of the misbehaved user/application to the admitted level, or the misbehaved user's acceptance of performance degradation. This may also result in degradation of performance for other users of the workstation which should be omitted by the OS control mechanisms. If host QoS changes result in degradation of the application performance, the host renegotiation entity may invoke the resource administration protocol to lower the QoS parameters in the network between the sender and the receiver.

5.1.3 Network Request for Renegotiation/Change

Overload of the network at some nodes can cause a renegotiation request for QoS change. This request comes as a notification from the resource administration protocol to the host reporting that the allocation of resources has to change. There are two possibilities: (1) the network can adapt to the overload; or (2) the network cannot adapt to the overload. In the former case, the network still needs to notify the host because some degradation may occur during the modification of resources (e.g., if the network tears down a connection and establishes a new connection). This actually may interrupt the media flow, so the host has to react to this change. In the latter case, the source

(host) must adapt.

5.2 Adaptive Schemes for Network Change

We describe several adaptation mechanisms for resource adaptation when the renegotiation request comes from the network due to network overload. The adaptation mechanisms implicitly offer partial solutions for cases when the renegotiation request comes from the user or from the host system.

5.2.1 Network Adaptation

The fixed routing and resource reservation for each conversation, combined with load fluctuations, introduce problems of network unavailability and loss of network management. Thus, a proper balancing of the network load is desirable and necessary in order to (1) increase network availability; (2) allow network administrators to reclaim resources; and (3) reduce the impact of unscheduled, run-time maintenance on clients with guaranteed services. Efficient routing and resource allocation decisions, made for previous clients which made requests for QoS guarantees, reduce the probability that a new client's request will be refused by the admission scheme. The more efficient the routing and the resource allocation is, the greater the number of guaranteed connection requests is accepted.

One possibility for implementing a *load balancing policy* is to employ the following mechanisms: *routing*, *performance monitoring* (detecting load changes), *dynamic re-routing* (changing the route), *load balancing control* (making a decision to re-route a channel) [13]. The routing mechanism implements the routing algorithm which selects a route in adherence to certain routing constraints. The performance monitoring mechanism monitors the appropriate network performance and reports them to the load balancing control. The dynamic re-routing mechanism is needed to establish the alternative route and to perform a transparent transition from the primary route to the alternative route. The load balancing control mechanism receives information from the the performance monitoring mechanism and determines whether load balancing can be attempted using a load balancing algorithm defined by the policy. If load balancing can be attempted, the routing mechanism provides an alternative route and the transition from the primary route to the alternative route is accomplished using the dynamic re-routing mechanism.

The adaptive resource scheme in this protocol is the *dynamic re-routing mechanism*. When channel i is to be re-routed, the source tries to establish a new channel that has the same traffic and performance parameters and shares the same source and destination as channel i, but takes a different route. The new channel is called *a shadow channel* [13] of a channel i. After the shadow

channel has been established, the source can switch from channel i to the shadow channel and start sending packets on it. After waiting for the maximum end-to-end delay time of channel i, the source initiates a tear-down message for channel i. If the shadow channel shares part of the route with the old channel, it is desirable to let the two channels share resources. This further implies that the establishment and tear-down procedures are aware of this situation, so that the establishment does not request the new resource and the tear-down procedure does not free the old resource.

5.2.2 Source Adaptation

Another alternative reaction to changes in the network load is to adapt the source rate according to the currently available network resources. This approach requires *feedback* information from the network to the source which results in graceful degradation in the media quality during periods of congestion. For example, in [10] the feedback control mechanism is based on predicting the evolution of the system state over time. The predicted system state is used to compute the target sending rate for each frame of video data. The adaptation policy strives to keep the bottleneck queue size for each connection at a constant level. Each switch monitors the buffer occupancy and the service rate per connection. The buffer occupancy information is a count of the number of queued packets for the connection at the instant when the feedback message is sent. The rate information is the number of packets transmitted for the connection in the time interval between two feedback messages. There are two possibilities to implement the feedback transmission mechanism. (1) The per-connection state information is periodically appended to a data packet for the corresponding connection. At the destination, this information is extracted and sent back to the source. A switch updates the information fields in a packet only if the local service rate is lower than that reported by a previous switch along the path. (2) The feedback message is sent in a separate control packet which is sent back along the path of the connection towards the source.

Other source adaptation schemes (for video traffic) may control overload. (1) *Rate control using network feedback.* In this approach each source adapts to changes in network conditions caused by an increase or decrease in the number of connections or by sudden changes in the sending rates of existing connections. Changes in the traffic conditions are detected by explicit or implicit feedback from the network. Explicit feedback is in the form of information about the traffic loads or buffer occupancy levels. Implicit feedback information about packet losses and round robin delay is available from acknowledgments. (2) *Traffic shaping at source.* Another way to control congestion is to smooth out traffic at the source. Typically most of the burstiness reduction can be obtained

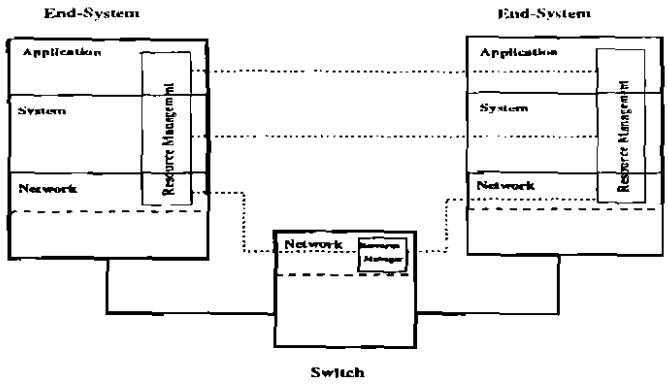


Figure 13: *Resource Management in the Networked Multimedia Systems*

by smoothing over an interval of 1-4 frames [10]. (3) *Hierarchical coding.* Hierarchical coding describes algorithms which produce two or more types of cells describing the same block of pixels with different degree of detail. However, these coders are more complex and use a greater amount of bandwidth to transmit images than single-layer coders [10].

6 Conclusion

Multimedia networked systems work in *connection-oriented mode*, although the Internet is an example of a connection-less network where QoS is introduced on a packet basis (every IP packet carries type of service parameters, because Internet does not have service notion). MNS, based on Internet protocol stack, uses RSVP, the new control reservation protocol, which accompanies the IP protocol and provides some kind of 'connection' along the path where resources are allocated.

QoS description, distribution, provision aud connected resource admission, reservation, allocation and provision, have to be embedded in different components of the multimedia communication architecture. This means that proper services and protocols in the end-points and the underlying network architectures have to be provided. Figure 13 shows that resource management is an integral part of the whole multimedia networked system.

The *end-point architectures* (Figure 1) need to incorporate components like resource managers for end-point resources and service agents for communication with network management. Resource managers need to include translation services, admission control, resource reservation and management for the end-point. Further, resource managers and service agents need access to MIBs with QoS specifications, which can be used by the resource administration (for status reporting) of the resources. Especially, the system domain needs to have QoS and resource management. Several important issues, as described in detail in previous sections, have to be considered in the

end-point architectures: (1) QoS specification, negotiation, and provision; (2) resource admission and reservation for end-to-end QoS; and (3) QoS configurable transport systems.

The *network routers/switches* need to employ MIBs, resource managers, and network management to provide connections/VCIs/channels with QoS guarantees. Further, resource managers must consist of several components such as, for example, packet classifier and packet scheduler for QoS provision, as well as admission controller for admission of resource. The network management includes traffic monitors in the form of agents which communicate in order to have a global view on network resources.

Some examples of architectural choices where QoS and resource management is designed and implemented are enumerated:

1. The OSI architecture provides QoS in the network layer and some enhancements in the transport layer. The OSI 95 project considers integrated QoS specification and negotiation in the transport protocols [11].

2. The Lancaster's QoS-Architecture (QoS-A) [3] offers a framework to specify and implement the required performance properties of multimedia applications over high-performance ATM-based networks. The QoS-A incorporates the notions of *flow*, *service contract* and *flow management*. The *multimedia enhanced transport service (METS)* provides the functionality to contract QoS.

3. The Heidelberg Transport System (HeiTS) [4], based on ST-II network protocol, provides continuous-media exchange with QoS guarantees, upcall structure, resource management and real-time mechanisms. HeiTS transfers continuous media data streams from one origin to one or multiple targets via multicast. HeiTS nodes negotiate QoS values by exchanging flow specification to determine the resources required - delay, jitter, throughput and reliability.

4. The UC Berkeley's Tenet Protocol Suite with protocol set RCAP, RTIP, RMTP, CMTP provides network QoS negotiation, reservation, and resource administration through the RCAP control and management protocol.

5. Internet protocol stack, based on IP protocol, provides resource reservation if RSVP control protocol [9] is used.

6. QoS handling and management is provided in UPenn's end-point architecture at the application and transport subsystems [2], where the *QoS Broker*, as the end-to-end control and

management protocol, implements QoS handling over both subsystems and relies on control and management in ATM networks.

Resource management, based on QoS requirements, has become an important part of multimedia communication systems across all system components because of the requests for resource guarantees. These requests are introduced because of: (1) increasing application varieties and requirements; and (2) incoming new media transmitted over the high-speed networks.

Acknowledgment

Klara Nahrstedt's work was supported by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, as well as by Bell Communications Research under Project DAWN, by an IBM Faculty Development Award to Prof. Jonathan M. Smith, and by the Hewlett-Packard Company. The authors gratefully acknowledge many valuable ideas and suggestions from Jonathan M. Smith. Thank you!

References

[1] Ralf Steinmetz, "Multimedia Operating Systems: Resource Reservation, Scheduling, File Systems, and Architectures", *Technical Report*, No.43.9402, IBM European Networking Center, Heidelberg, Germany, March, 1994

[2] Klara Nahrstedt, Jonathan M. Smith, "The QoS Broker", *Technical Report*, MS-CIS-94-13, University of Pennsylvania, March 1994, Submitted to ACM Multimedia '94

[3] 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

[4] Lars C. Wolf, Ralf Guido Herrtwich, "The System Architecture of the Heidelberg Transport System", *Technical Report*, No. 43.9319, IBM European Networking Center, Heidelberg, Germany, 1993

[5] C. Brendan S. Traw and Jonathan M. Smith, "Hardware/Software Organization of a High-Performance ATM Host Interface," in *IEEE JSAC (Special Issue on High Speed Computer/Network Interfaces)*, pp. 240-253, February 1993.

[6] Jonathan M. Smith and C. Brendan S. Traw, "Giving Applications Access to Gb/s Networking", *IEEE Network*, pp. 44-52, July 1993

[7] Ralf Steinmetz, Clemens Engler, "Human perception of Media Synchronization", *Technical Report*, No. 43.9310, IBM European Networking Center, Heidelberg, Germany, 1993

[8] Bert J. Dempsey, Jorg Liebeherr, Alfred C. Weaver, "A New Error Control Scheme for Packetized Voice over High-Speed Local Area Networks", *Proceedings of 18th Conference on Local Computer Networks*, Minneapolis, MN,September, 1993

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

[10] Hemant Kanakia, Partho P. Mishra, Amy Reibman, "An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport", *Proceedings SIGCOMM '93*, Baltimore, August, 1993

[11] Andre Danthine, Oliver Bonaventure, "From Best Effort to Enhanced QoS", *Technical Report*, No. R2060/ULg/CIO/DS/P/004/b1, University of Liege, Belgium, July 1993

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

[13] Colin Parris, Hui Zhang, Domenico Ferrari, "A Mechanism for Dynamic Re-routing of Real-Time Services on Packet Networks", *Technical Report*, UC Berkeley, CA, 1992

[14] L. Sha, M. H. Klein, J. B. Goodenough, "Rate Monotonic Analysis for Real-Time Systems", *Foundations of Real-Time Computing, Scheduling and Resource Management*; Kluwer Academic Publisher; Norwell, 1991; pp. 129-156.

[15] Hui Zhang, Srinivasan Keshav, "Comparison of Rate-Based Service Disciplines", *Proceedings SIGCOMM'91*, Zurich, Switzerland, September, 1991

[16] A. Banerjea, B. Mah, "The Real-Time Channel Administration Protocol", *Proceedings Second International Workshop on Network and Operating System for Digital Audio and Video*, Heidelberg, Germany, November, 1991

[17] L. Sha, R. Rajkumar, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", *IEEE Transactions on Computers*, Vol.3, No.9, 1990

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