Chapter 11

Multimedia Communication Systems

The consideration of multimedia applications supports the view that local systems expand toward distributed solutions. Applications such as kiosks, multimedia mail, collaborative work systems, virtual reality applications and others require high-speed networks with a high transfer rate and communication systems with adaptive, lightweight transmission protocols on top of the networks.

In Chapter 10, high-speed network requirements, such as low latency for interactive operations, high bandwidth above 10 Mbps, and low error-bit rate were discussed. Several networks were described that support transmission of multimedia, such as FDDI, High-Speed Ethernet, Priority Token Ring (PRT) or ATM networks with bandwidth of 100 Mbps and above.

In this chapter we discuss important issues related to multimedia communication systems above the data link layer. From the communication perspective, we divide the higher layers of the Multimedia Communication System (MCS) into two architectural subsystems: an *application subsystem* and a *transport subsystem.*

In the section on application subsystems, management and service issues for group collaboration and session orchestration are presented. Group collaboration and session management provide support for a large group of multimedia applications, such

as tele-collaboration. The section on transport subsystems includes a presentation of transport and network layer protocols that are used for the standardized support of networked multimedia applications. The third section of this chapter consists of a discussion about *Quality of Service* (QoS) and resource management in MCSs.

11.1 Application Subsystem

11.1.1 Collaborative Computing

The current infrastructure of networked workstations and PCs, and the availability of audio and video at these end-points, makes it easier for people to cooperate and bridge *space and time*. In this way, network connectivity and end-point integration of multimedia provides users with a *collaborative computing* environment. Collaborative computing is generally known as *Computer-Supported Cooperative Work* (CSCW).

There are many tools for collaborative computing, such as *electronic mail*, *bulletin boards* (e.g., Usenet news), *screen sharing tools* (e.g., ShowMe from SunSoft), *text-based conferencing systems* (e.g., Internet Relay Chat, CompuServe, American Online), *telephone conference systems*, *conference rooms* (e.g., VideoWindow from Bellcore), and *video conference systems* (e.g., MBone tools nv, vat). Further, there are many implemented CSCW systems that unify several tools, such as Rapport from AT&T, MERMAID from NEC and others.

In this section we present a framework for collaborative computing and general related issues exemplified by different systems and tools.

Collaborative Dimensions

Electronic collaboration can be categorized according to three main parameters: *time*, *user scale* and *control* [WSM+91]. Therefore, the collaboration space can be partitioned into a three-dimensional space (shown in Figure 11.1).

- *Time*

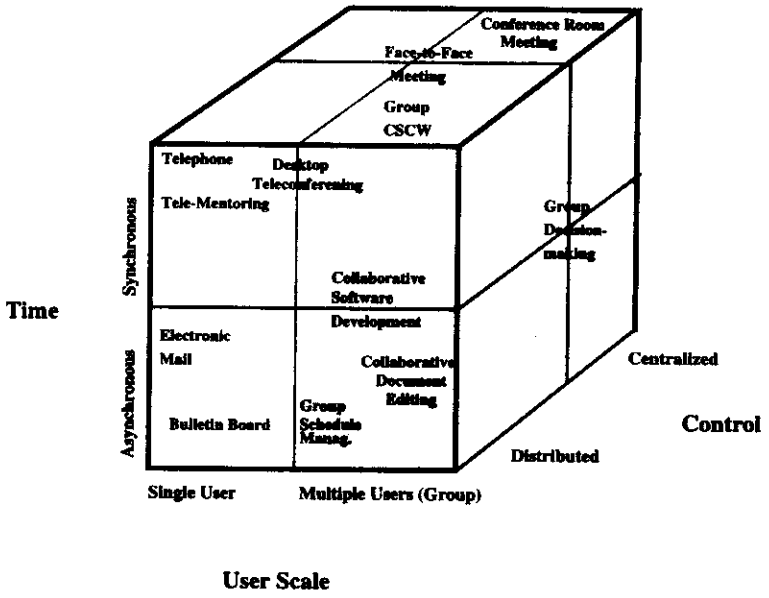

Figure 11.1: *Dimensions of collaborative computing.*

With respect to time, there are two modes of cooperative work: *asynchronous* and *synchronous*. *Asynchronous cooperative work* specifies processing activities that do not happen at the same time; the *synchronous cooperative work* happens at the same time.

- *User Scale*

  The user scale parameter specifies whether a *single user* collaborates with another user or a *group* of more than two users collaborate together. Groups can be further classified as follows [Stu94]:

  - A group may be *static* or *dynamic* during its lifetime. A group is *static* if its participating members are pre-determined and membership does not change during the activity. A group is *dynamic* if the number of group members varies during the collaborative activity, i.e., group members can join or leave the activity at any time.
  - Group members may have different roles in the CSCW, e.g., a *member* of a group (if he or she is listed in the group definition), a *participant* of a

group activity (if he or she successfully joins the conference), a *conference initiator*, a *conference chairman* , a *token holder* or an *observer.*

- Groups may consist of members which have *homogeneous* or *heterogeneous* characteristics and requirements of their collaborative environment.

- *Control*

  Control during a collaboration can be *centralized* or *distributed. Centralized control* means that there is a chairman (e.g., main manager) who controls the collaborative work and every group member (e.g., user agent) reports to him or her. *Distributed control* means that every group member has control over his/her own tasks in the collaborative work and distributed control protocols are in place to provide consistent collaboration.

Other partition parameters may include *locality, and collaboration awareness. Locality* partition means that a collaboration can occur either in the *same place* (e.g., a group meeting in an office or conference room) or among users located in *different places* through *tele-collaboration.* With respect to this parameter, we assume in this section tele-collaboration.

Collaboration awareness divides group communication systems into *collaboration-transparent*, and *collaboration-aware* systems. The collaboration-transparent system is an existing application (e.g., a favorite text processor/spreadsheet) extended for collaboration. For example, some new *document editing* systems are collaboration-transparent because single user document editors were expanded for simultaneous editing of a shared document among several users. The collaborative-aware system is a dedicated software application for CSCW. For example, a conferencing system is a collaborative-aware system.

Group communication systems can be further categorized into *computer-augmented collaboration* systems, where *collaboration* is emphasized, and *collaboration-augmented computing* systems, where the concentration is on *computing* [MR94]. Computer-augmented collaboration is centered around a social activity, for instance discussion, or decision-making, where the computers and networks help to improve this activity. Collaboration-augmented computing is centered around tools that accommodate multiple users.

Group Communication Architecture

Group communication (GC) involves the communication of multiple users in a synchronous or an asynchronous mode with centralized or distributed control (the gray area in Figure 11.1).

A group communication architecture consists of a *support model, system model* and *interface model* [WSM+91]. The GC *support model* includes *group communication agents* that communicate via a multi-point multicast communication network as shown in Figure 11.2. Group communication agents may use the following for their

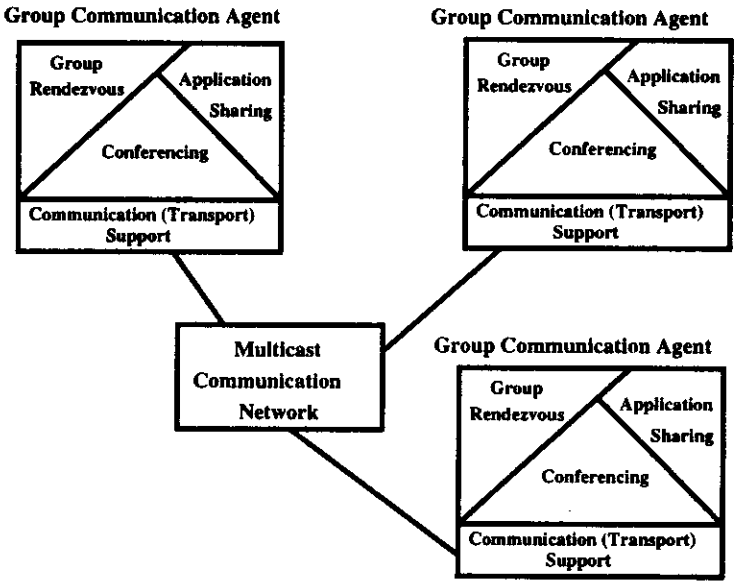

Figure 11.2: *Group communication support model.*

collaboration:

- *Group Rendezvous*

  Group rendezvous denotes a method which allows one to organize meetings, and to get information about the group, ongoing meetings and other static and dynamic information.

- *Shared Applications*

Application sharing denotes techniques which allow one to replicate information to multiple users simultaneously. The remote users may point to interesting aspects (e.g., via tele-pointing) of the information and modify it so that all users can immediately see the updated information (e.g., joint editing). Shared applications mostly belong to collaboration-transparent applications.

- *Conferencing*

  Conferencing is a simple form of collaborative computing. This service provides the management of multiple users for communicating with each other using multiple media. Conferencing applications belong to collaboration-aware applications.

The GC *system model* is based on a client-server model. Clients provide user interfaces for smooth interaction between group members and the system. Servers supply functions for accomplishing the group communication work, and each server specializes in its own function. For example, the MONET system provides a conference server for conferencing capabilities, an application sharing server for providing a *shared or group workspace*, a directory server for group rendezvous services (directory and registration services) and a multimedia server for intermedia synchronization [SRB+92].

The GC *interface model* includes two kinds of protocols for exchanging information within the GC support model: *user presentation protocols* and *group work management protocols*. User presentation protocols perform interactions among the clients, such as opening a conference, closing a conference, dynamic joining and leaving of a meeting and floor passing. Group work management protocols specify the communication between the clients and the servers. Services such as registration of active conferences and queries for further conference information are supported by these protocols.

Group Rendezvous

Group rendezvous methods allow for setting up collaborative group meetings and providing other static and dynamic information about groups and ongoing or future

meetings. The group rendezvous tools provide a single set of session (e.g., conference) and activity information at the user interface. There are *synchronous* and *asynchronous methods* for group rendezvous:

- *Synchronous Rendezvous Methods*

  Synchronous rendezvous methods use *directory services* and *explicit invitations*. *Directory services* (e.g., X.500) access information stored in a knowledge base about the conference, such as the name of the conference, registered participants, authorized users and name and role of the participants. Examples of conferencing systems which use the directory method for group rendezvous are:

  - MBone's session directory tool, *sd*

    sd may be used as a guide to announce open conferences [JM93a]. *sd* resides at a known address and port on each user's workstation and both listen for announcements and post their own sessions.

  - Touring machine's *name server query*

    The name server acts as a central repository of both static and dynamic information, such as authorized users, registered clients and ongoing sessions [Lab93]. A client can query for all interesting sessions to join or all clients in a particular session.

  - MONET's *directory service*

    The directory server provides directory and registration services. A catalog of various resources such as people, machines and tools across the network is available to the directory service. A registration service deals with registering participants and groups for conferences [SRB+92].

  The *explicit invitations* method sends invitations either point-to-point or point-to-multipoint to conference participants. The problem here is that calling others requires the initiator of the conference to know where users reside. An example of group rendezvous supported via explicit invitations, is ISI's (USC Information Sciences Institute) session orchestration tool, *mmcc* [SW94a].

- *Asynchronous Rendezvous Methods*

Asynchronous rendezvous methods may be implemented through *e-mail* or *bulletin boards*. Borenstein suggests *e-mail* as a platform for group rendezvous, embedded in synchronous conferencing applications [Bor92]. The e-mail-based mechanism encapsulates in the body message enough information about a group session establishment. This scheme builds on the already existing e-mail infrastructure for both distributing information and addressing end-users.

Bulletin boards are used to support asynchronous rendezvous. Local bulletin boards on the Internet already announce seminars, classes, conferences and other open meetings of a school or institution. The *World Wide Web* (WWW) offers new possibility for global group rendezvous offers. The WWW infrastructure is beginning to be used to announce and dynamically update public sessions or upcoming open conferences [IET94].

Application Sharing Approach

Sharing applications is recognized as a vital mechanism for supporting group communication activities. Sharing applications means that when a shared application program (e.g., editor) executes any input from a participant, all execution results performed on the *shared object* (e.g., document text) are distributed among all the participants. Shared objects are displayed, generally, in *shared windows* [HTM92].

Application sharing is most often implemented in collaboration-transparent systems, but can also be developed through collaboration-aware, special-purpose applications. An example of a software toolkit that assists in development of shared computer applications is Bellcore's *Rendezvous* system (language and architecture) [HBP+93]. Shared applications may be used as conversational props in tele-conferencing situations for collaborative document editing and collaborative software development.

An important issue in application sharing is *shared control*. The primary design decision in sharing applications is to determine whether they should be *centralized* or *replicated* [OMS+92, SW94a]:

- *Centralized Architecture*

  In a centralized architecture, a single copy of the shared application runs at

one site. All participants' input to the application is forwarded to the local site and the application's output (shared object) is then distributed to all sites. Figure 11.3 (a) shows the centralized architecture.

The advantage of the centralized approach is *easy maintenance* because there is only one copy of the application that updates the shared object. The disadvantage is *high network traffic* because the output of the application needs to be distributed every time. In the case of images, the output data traffic may be significant. This means that a *high-bandwidth network* is needed.

- *Replicated Architecture*

  In a replicated architecture, a copy of the shared application runs locally at each site. Input events to each application are distributed to all sites and each copy of the shared application is executed locally at each site. Figure 11.3 (b) shows the replicated architecture.

  The advantages of this architecture are *low network traffic*, because only input events are distributed among the sites, and *low response times*, since all participants get their output from local copies of the application. The disadvantages are the requirement of *the same execution environment* for the application at each site, and the *difficulty in maintaining consistency.* The problem is discussed below.

As stated above, one important problem with shared applications in a replicated architecture is maintaining the *consistency of shared objects.* A variety of mechanisms exist to maintain data consistency among group members. Examples include *centralized locks, floor passing schemes, and dependency detections.* Here, we discuss *floor passing control* because this kind of control is the most relevant to group communication.

The notion of a *floor* is used to maintain the consistency of shared object *data* (multimedia documents) or *applications* (programs) shared among participants. The group member who holds the floor (the *floor holder*) has the right to manipulate shared objects in shared windows. Hence, the floor holder can perform operations such as opening and closing shared windows, loading documents into a shared window and issuing an input event to the shared application to manipulate shared data.

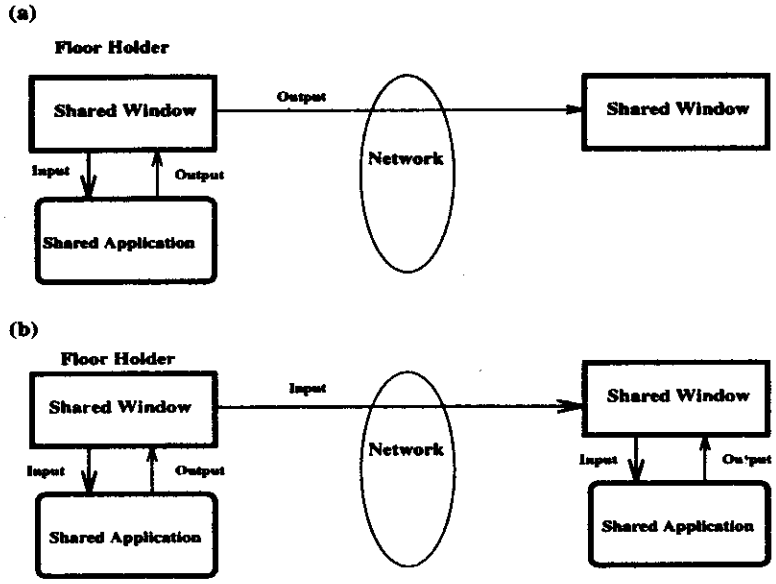

Figure 11.3: *Sharing application architectures: (a) centralized architecture, (b) replicated architecture.*

A possible shared data manipulation architecture is shown in Figure 11.4. A CSCW control component resides at every site and dispatches input events coming from an *input* device (e.g., keyboard). It checks if the active site is a floor holder. If the site is a floor holder, it accepts and processes the input event as well as distributes the input event to other sites. If the site is not a floor holder, the CSCW control discards its own input and accepts input events coming from another site.

A replicated shared control architecture is used in CSCW systems such as MERMAID (Multimedia Environment for Remote Multiple Attendee Interactive Decision-making) from NEC [OMS+92], the broadband ISDN group tele-working system from Hitachi [HTM92] and others.

Conferencing

Conferencing supports collaborative computing and is also called *synchronous tele-collaboration.* Conferencing is a *management service* that controls the communica-

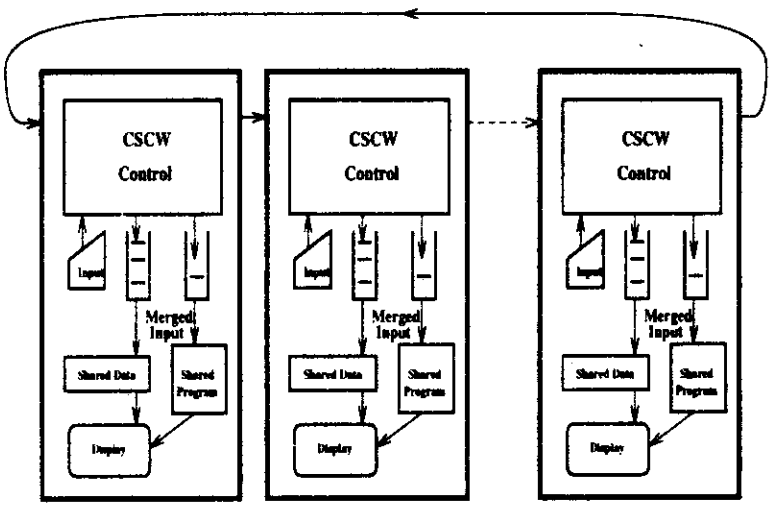

Figure 11.4: *Shared data manipulation architecture.*

tion among multiple users via multiple media, such as video and audio, to achieve simultaneous face-to-face communication. More precisely, video and audio have the following purposes in a tele-conferencing system:

- *Video* is used in technical discussions to display view-graphs and to indicate how many users are still physically present at a conference. For visual support, workstations, PCs or *video walls* can be used.

  For conferences with more than three or four participants, the screen resources on a PC or workstation run out quickly, particularly if other applications, such as shared editors or drawing spaces, are used. Hence, mechanisms which quickly resize individual images should be used.

  This situation also implies that large *video walls* in conference rooms with multiple, high-resolution screens (e.g., Bellcore's VideoWindow [FKRR93]) may continue to be necessary even after workstations are video-equipped everywhere. The systems supporting conference rooms with video walls attempt to provide the richness of human communication situations, such as unplanned hallway encounters (e.g., Cruiser from Bellcore), drop-in seminars, panel discussions (e.g., Media Space from Xerox PARC), jury trials and lectures.

- *Audio* is an important component in tele-conferencing for describing and clarifying visual information. Therefore, quality audio, with true full-duplex communication and echo cancellation, and possibly enhanced with spatial queues, is necessary.

Conferencing services rely on low network latency for acceptable user interactivity and high bandwidth for potentially data-intensive media. Further, they rely on *distributed messaging* for transmission of data and/or control information.

Conferencing services control a *conference* (i.e., a collection of shared state information such as *who is participating in the conference, conference name, start of the conference, policies associated with the conference*, etc.) *Conference control* includes several functions:

- *Establishing* a conference, where the conference participants agree upon a common state, such as identity of a chairman (moderator), access rights (floor control) and audio encoding. Conference systems may perform *registration, admission*, and *negotiation* services during the conference establishment phase, but they must be flexible and allow participants to join and leave individual media sessions or the whole conference. The flexibility depends on the control model.
- *Closing* a conference.
- *Adding* new users and *removing* users who leave the conference.

Conference control utilizes and cooperates with per-media session management components (see Section 11.1.2.) to tie together several streams.

Conference states can be stored (located) either on a central machine (*centralized control*), where a central application acts as the repository for all information related to the conference, or in a *distributed* fashion. The control model follows from the location of the conference state. Accordingly, the control model may be either *centralized* or *distributed.*

- *Centralized Conference Control*

Centralized conference control provides the establishment of a conference. First, the initiator (e.g., chairman) starts a conference by selecting an initial group of invited conference members (explicit invitation for group rendezvous). This implies that the chairman knows the addresses of all conference participants. The knowledge of the conference state is inquired from a central directory server, which implies that the client has registered his/her location.

Second, each invited client *responds* to the invitation so that the initiator is informed of who will participate in the conference. After this step, a *negotiation* of conference policies and an *admission* of resources is performed among the conference participants. During the negotiation, the shared conference state is distributed using a *reliable messaging* service to all conference participants. All information related to the conference is stored on a central machine.

This *static control*, implemented through explicit exchange of the conference state, guarantees the consistency of the state space to every participant, and works well for small conferences.

The advantage of the centralized conference control is the guaranteed *consistency* of the conference state. The disadvantage is that when a new participant (outside of the invited group) wants to join, explicit exchange of the conference state must be performed among all participants, which causes large delays. Furthermore, if a conference participant has a link failure, it is more difficult to re-establish the conference state.

- *Distributed Conference Control*

  Distributed conference control is based on a *distributed conference state*. This is achieved as follows: the initiator of the conference establishes a multicast space (e.g., multicast tunnel in MBone) with multicast entries for distribution of information to the conference participants and the conference is established; the conference participants join the conference by tuning into a particular multicast entry (e.g., multicast address), announced through group rendezvous means (e.g., sd).

  Each site distributes its own participation status to other conference participants, but there is no global notion of a group membership, and no guarantees that all users will have the same view of the state space. Hence, this *loose*

control is implemented through retransmitting the state periodically for eventual consistency. The periodical retransmission is done using an *unreliable messaging* service. The loose control works well for large conferences. This loose control is provided by sessions called *lightweight sessions* [JMF93], which are used on the *Multicast Backbone (MBone)* [MB94]. MBone is a multicast-capable segment of Internet which has been used for a number of applications including multimedia (audio, video and shared workspace) conferencing. These applications include *vat* (LBL's Visual Audio Tool) [JM92], *ivs* (INRIA Videoconferencing System) [INR93], *nv* (Xerox's Network Video tool) [Fre92] and *wb* (LBL's shared whiteboard) [JM93b].

Advantages of distributed conference control are: *inherent fault tolerance*, which means that if a network connection breaks down in the middle of a conference and it is repaired, it is easier to re-establish the shared conference state since there is no strict consistency requirement; and *scaling* properties, although at some point refresh periodicity needs to adapt to the size and scope of the conference, otherwise, the conference may be in danger of flooding itself with session reports. The disadvantage is that the conference participants may not have the same view of the state space.

Given the wide diversity of *collaboration styles*, it appears difficult to derive a canonical conference control protocol, although there are some attempts to achieve this goal with the *Conference Control Channel Protocol (CCCP)*[HW94].

It is promising to develop common, underlying control functions and allow the combination of these in appropriate ways [Sch94a]. Further, *state agreement protocols* might be invoked to agree on the shared state, also called the *ephemeral teleconferencing state* [SW94b]. The state is *ephemeral* because the importance of the state is valid only during the duration of the conference. Agreement Protocols with distributed control involve a *policy* to control the session state. In [SW94b], three aspects of policies are identified: *initiator of policies*, *voting policies* and *consistency policies*. The first aspect specifies which members may initiate certain change operations. The voting policy specifies the decision about the change of the shared state which was issued by a group member. Modification of the shared state is based on *voting rules*. To achieve consistency, the state agreement covers the functionality of the floor and access control, media negotiation, directory services for conferences,

invitation services, user location services, etc.

11.1.2 Session Management

Session management is an important part of the multimedia communication architecture. It is the core part which separates the control, needed during the transport, from the actual transport. Session management is extensively studied in the collaborative computing area, therefore we concentrate on architectural and management issues in this area.

Architecture

A session management architecture is built around an entity – *session manager* – which separates the control from the transport [SC92]. By creating a reusable session manager, which is separated from the user-interface, conference-oriented tools avoid a duplication of their effort. A possible session control architecture is shown in Figure 11.5. The session control architecture consists of the following components:

- *Session Manager*

  Session manager includes *local* and *remote* functionalities. Local functionalities may include (1) *membership control management*, such as participant authentication or presentation of coordinated user interfaces; (2) control management for shared workspace, such as *floor control*; (3) *media control management*, such as intercommunication among media agents or synchronization; (4) *configuration management*, such as exchange of interrelated QoS parameters or selection of appropriate services according to QoS; and (5) *conference control management*, such as an establishment, modification and a closing of a conference. Remotely, the session manager communicates with other session managers to exchange session state information which may include the floor information, configuration information, etc. It is important to note that in different conferencing systems, the conference control and floor control can be embedded either in the application layer (e.g., Group Tele-Working System from Hitatchi labs) or in the session layer (e.g., Touring machine from

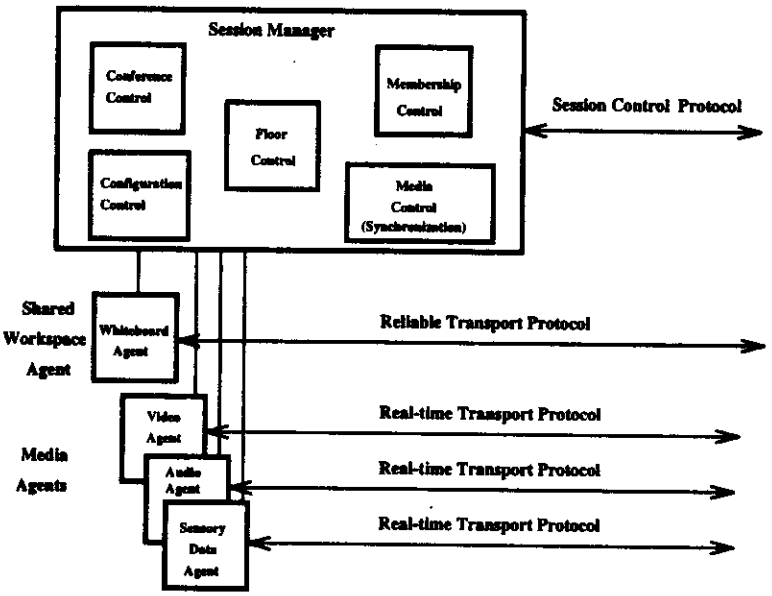

Figure 11.5: *Session control architecture.*

Bellcore).

- *Media Agents*

  Media agents are separate from the session manager and they are responsible for decisions specific to each type of media. This modularity allows a replacement of agents. Each agent performs its own control mechanism over the particular medium, such as mute, unmute, change video quality, start sending, stop sending, etc.

- *Shared Workspace Agent*

  The shared workspace agent transmits shared objects (e.g., telepointer coordinate, graphical or textual object) among the shared applications.

Control

Each session is described through its *session state.* This state information (start time of session, policies associated with the session, session name) is either *pri-*

vate (e.g., local resources) or *shared* among all session participants (e.g., conference participants).

Session management includes two steps to process the session state: an *establishment* and a *modification* of the session. During the establishment, the session manager negotiates, agrees, and sets the logical state of its own session. Further, it negotiates, agrees and sets billing policy and other policies with other session managers. Further, it permits "publishing" a session (e.g., using group rendezvous service), allowing others to locate and joint a session. It negotiates and defines the transport topology with the transport subsystem.

Dependent on the functions, which an application requires and a session control provides, several control mechanisms are embedded in session management:

- *Floor Control*

  Within shared workspaces, the *floor control* is employed to provide access to the shared workspace. Further, the floor control in shared applications (Section 11.1.1) is often used to maintain data consistency (social protocol). Each system makes decisions, such as the level of *simultaneity* and *granularity* at which to enforce access control. In the simplest form of floor control, applications use a *floor-passing mechanism* (gavel-passing [SW94a], chalk-passing [Sch94a]). The floor-passing mechanism means that at any time, only one participant has the floor. The floor is handed off to another participant when requested. To obtain the floor, the participant must explicitly take action to signal a floor change.

  With real-time audio, there is no notion of data consistency, instead, the floor control is typically used in more formal settings to promote turn-taking (e.g., distributed classroom). For more life-like audio interaction, no floor control is optimal.

  Floor control for real-time video is frequently used to control bandwidth usage.

  The floor control mechanisms are low-level means used to implement *floor policies* [RSV94]. A floor policy describes how participants request the floor and how the floor is assigned and released. Typically, a session-wide floor holder is selected by a chairman, although it is desirable to have flexible floor

control policies.

- *Conference Control*

  For conferencing applications, conference control is employed. We discussed a possible conference control (the centralized approach with static control versus the distributed approach with loose control) in Section 11.1.1).

- *Media Control*

  Media control mainly includes a functionality, such as the synchronization of media streams, which is broadly discussed in Chapter 15.

- *Configuration Control*

  Configuration control includes a control of media quality, QoS handling, resource availability and other system components to provide a session according to users requirements. QoS handling and corresponding resource management are discussed in Section 11.3. This control may embed services, such as the negotiation and renegotiation of media quality.

- *Membership Control*

  Membership control may include services, for example, *invitation* to a session, *registration* into a session, *modification* of the membership during the session, etc.

For distribution of the shared session control information among the session managers, *reliable messaging services* or *unreliable messaging services* with periodic refreshment can be used. The goal is to provide a *distributed messaging* service with different degrees of reliable delivery. For example, periodic update messages may not require reliable delivery at all. On the other hand, for floor exchange, it may be critical to know that the update was received by all participants, since if not, multiple video channels may, for example, result and may overextend the capacity of the network. There may even be conference participants who value the delivery of the same message type differently [HW94].

11.2 Transport Subsystem

We present in this section a brief overview of transport and network protocols and their functionalities, which are used for multimedia transmissions. We evaluate them with respect to their suitability for this task.

11.2.1 Requirements

Distributed multimedia applications put new requirements on application designers, as well as network protocol and system designers. We analyze the most important enforced requirements with respect to the multimedia transmission:

User and Application Requirements

Networked multimedia applications by themselves impose new requirements onto data handling in computing and communications because they need (1) substantial data throughput, (2) fast data forwarding, (3) service guarantees, and (4) multicasting.

- *Data Throughput*

  Audio and video data resemble a stream-like behavior, and they demand, even in a compressed mode, high *data throughput.* In a workstation or network, several of those 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*

  Fast data forwarding imposes a problem on end-systems where different applications exist in the same end-system, and they each require data movement ranging from normal, error-free data transmission to new time-constraint traffic types traffic. But generally, the faster a communication system can transfer a data packet, the fewer packets need to be buffered. This requirement leads

to a 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-like application, such as video-on-demand, a delay of up to one second may be easily tolerated. In an opposite dialogue application, such as a videophone or videoconference, demand end-to-end delays lower than typically 200 msec inhibit a natural communication between the users.

- *Service Guarantees*

  Distributed multimedia applications need *service guarantees*, otherwise their acceptance does not come through as these systems, working with continuous media, compete against radio and television services. 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 their users because transmission over unreserved resources leads to dropped or delayed packets [DHVW93].

- *Multicasting*

  Multicast is important for multimedia-distributed applications in terms of sharing resources like the network bandwidth and the communication protocol processing at end-systems.

Processing and Protocol Constraints

Communication protocols have, on the contrary, some constraints which need to be considered when we want to match application requirements to system platforms.

A typical multimedia application does not require processing of audio and video to be performed by the application itself. Usually the data are obtained from a source (e.g., microphone, camera, disk, 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., to copy data directly from adapter-to-adapter, and the program merely sets the correct switches for the data flow by connecting sources to sinks. Hence, the application itself never really touches the data as is the case in traditional processing.

A problem with direct copying from adapter-to-adapter is the control and the change of QoS parameters. In multimedia systems, such an adapter-to-adapter connection is defined by the capabilities of the two involved adapters and the bus performance. In today's systems, this connection is static. This architecture of low-level data streaming corresponds to 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 [Fin91, HM91]. Note, in practice we encounter headers and trailers surrounding continuous-media data coming from devices and being delivered to devices. In the case of compressed video data, e.g., MPEG-2, the program stream contains several layers of headers compared with the actual group of pictures to be displayed.

Protocols involve a lot of *data movement* because of the layered structure of the communication architecture. But copying of data is expensive and has become a bottleneck, hence other mechanisms for buffer management must be found.

Different layers of the communication system may have different PDU sizes, therefore, a *segmentation* and *reassembly* occur. This phase has to be done fast, and efficient. Hence, this portion of a protocol stack, at least in the lower layers, is done in hardware, or through efficient mechanisms in software.

Some parts of protocols may use *retransmission error-recovery* mechanism 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* (most of them, although, for example, FDDI offers also an isochronous transfer mode which is best suited for multimedia transmission) put requirements on the protocol design for continuous media. What has to happen is that the higher protocols must provide a 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 (e.g., ST-II) where, during the connection establishment, preparation for "synchronous transmission" of continuous media has to occur, or some connection-like behavior has to be enforced to provide service guarantees (e.g., RSVP with IP).

11.2.2 Transport Layer

Transport protocols, to support multimedia transmission, need to have new features and provide the following functions: *timing information*, *semi-reliability*, *multicasting*, *NAK (None-AcKnowledgment)-based error recovery mechanism* and *rate control*.

First, we present transport protocols, such as TCP and UDP, which are used in the Internet protocol stack for multimedia transmission, and secondly we analyze new emerging transport protocols, such as RTP, XTP and other protocols, which are suitable for multimedia.

Internet Transport Protocols

The Internet protocol stack includes two types of transport protocols:

- *Transmission Control Protocol (TCP)*

  Early implementations of video conferencing applications were implemented on top of the TCP protocol. TCP provides a reliable, serial communication path, or virtual circuit, between processes exchanging a full-duplex stream of bytes. Each process is assumed to reside in an Internet host that is identified by an IP address. Each process has a number of logical, full-duplex ports through which it can set up and use as *full-duplex TCP connections*.

  Multimedia applications do not always require full-duplex connections for the transport of continuous media. An example is a TV broadcast over LAN, which requires a full-duplex control connection, but often a simplex continuous media connection is sufficient.

  During the *data transmission* over the TCP connection, TCP must achieve *reliable, sequenced delivery* of a stream of bytes by means of an underlying, unreliable datagram service. To achieve this, TCP makes use of retransmission on timeouts and positive acknowledgments upon receipt of information. Because retransmission can cause both out-of-order arrival and duplication of data, sequence numbering is crucial. Flow control in TCP makes use of a

window technique in which the receiving side of the connection reports to the sending side the sequence numbers it may transmit at any time and those it has received contiguously thus far.

For multimedia, the positive acknowledgment causes substantial overhead as all packets are sent with a fixed rate. Negative acknowledgment would be a better strategy. Further, TCP is not suitable for real-time video and audio transmission because its retransmission mechanism may cause a violation of deadlines which disrupt the continuity of the continuous media streams. TCP was designed as a transport protocol suitable for non-real-time reliable applications, such as file transfer, where it performs the best.

- *User Datagram Protocol (UDP)*

  UDP is a simple extension to the Internet network protocol IP that supports multiplexing of datagrams exchanged between pairs of Internet hosts. It offers only multiplexing and checksumming, nothing else. Higher-level protocols using UDP must provide their own retransmission, packetization, reassembly, flow control, congestion avoidance, etc.

  Many multimedia applications use this protocol because it provides to some degree the real-time transport property, although loss of PDUs may occur. For experimental purposes, UDP above IP can be used as a simple, unreliable connection for medium transport.

  In general, UDP is not suitable for continuous media streams because it does not provide the notion of connections, at least at the transport layer; therefore, different service guarantees cannot be provided.

Several extensions are proposed to increase the performance of both UDP and TCP protocols so that a larger group of applications (i.e., also multimedia applications) can use them. Large windows and time stamps are now draft standards. Also, selective acknowledgments might be taken in, although this mechanism provides a win when the packet loss rate approaches one per round-trip time [Bin93].

Real-time Transport Protocol (RTP)

RTP is an end-to-end protocol providing network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data over multicast or unicast network services. It is specified and still augmented by the Audio/Video Transport Working Group [SCFJ94]. RTP is primarily designed to satisfy the needs of multi-party multimedia conferences, but it is not limited to that particular application.

RTP has a companion protocol RTCP (RTP-Control Protocol) to convey information about the participants of a conference. RTP provides functions, such as *determination of media encoding, synchronization, framing, error detection, encryption, timing* and *source identification.* RTCP is used for the *monitoring of QoS* and for *conveying* information about the participants in an ongoing session. The first aspect of RTCP, the monitoring, is done by an application called a QoS monitor, which receives the RTCP messages. This monitor estimates the current QoS for monitoring, fault diagnosis and long-term statistics. The second aspect of RTCP is used for "loosely controlled" sessions. RTP and RTCP information is transmitted through separate ports.

RTP does not address resource reservation and does not guarantee QoS for real-time services. This means that it does not provide mechanisms to ensure timely delivery of data or guaranteed delivery, but relies on lower-layer services to do so. Further, it does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The RTP header carries sequence numbers to allow the end system to reconstruct the sender's packet sequence, but also they can be used to properly place the packet, for example, in video decoding.

RTP makes use of the network protocol ST-II or UDP/IP for the delivery of data. It relies on the underlying protocol(s) to provide demultiplexing.

Profiles are used to specify certain parts of the header for particular sets of applications. This means that particular media information is stored in an *audio/video profile*, such as a set of formats (e.g., media encodings) and a default mapping of those formats.

RTP is used in some of the MBone tools, for example, *nv*. The program *nv* is a packet video program used to support visual interaction in tele-conferencing over MBone. *nv* does all the network I/O, RTP the processing and the X window system does the interaction.

Xpress Transport Protocol (XTP)

XTP was designed to be an efficient protocol, taking into account the low error ratios and higher speeds of current networks [SDW92]. It is still in the process of augmentation by the XTP Forum to provide a better platform for the incoming variety of applications. XTP integrates transport and network protocol functionalities to have more control over the environment in which it operates [Che92]. XTP is intended to be useful in a wide variety of environments, from real-time control systems to remote procedure calls in distributed operating systems and distributed databases to bulk data transfer. It defines for this purpose six service types: *connection, transaction, unacknowledged data gram, acknowledged datagram, isochronous stream* and *bulk data.*

In XTP, the end-user is represented by a *context* becoming active within an XTP implementation. Two contexts (or several in multicast mode) are joined together to form an *association.* The path between two XTP sites is called a *route.* There are two types of XTP packets: information packets which carry user data, and control packets which are used for protocol management.

For *flow control*, XTP uses sliding window, or rate-based flow control. If window-based control is selected, the window size is negotiated during the connection setup. To advance the flow-control window, XTP uses a *combined mechanism* between a cumulative acknowledgment (e.g., TCP also uses a cumulative acknowledgement mechanism) and a selective acknowledgment, with a run-length encoding.

Data packet *retransmissions* are triggered by the arrival of status reports showing missing data. Status reports are requested and a certain timer controls the duration of the response to the request. After the timer expires and a status report was not received, a new status report is issued, and XTP enters a *synchronizing handshake*, where all further data transmission are halted until the correct status is received.

Therefore, XTP will never retransmit a data packet without positive indication that it has not been received.

The error management is different for each of these service types. Therefore, XTP error control is primary a set of building blocks, known as *mechanisms*, from which a variety of error control *policies* can be constructed. Therefore, error control features can be tailored to the needs of the user.

There are some features which meet the requirements for multimedia communication, such as [Mil93b]:

- XTP provides a connection-oriented transport and network transmission, hence it gives the benefit to map XTP on ATM networks and to use the possibilities of bandwidth reservation of ATM networks.
- Different transport services are provided: connection-mode, connectionless-mode and transaction-mode. Very important is the fast-connect-establishment for tele-transaction service.
- Flexible error management allows the turning off of the retransmission mechanism, which is useful for multimedia applications.
- XTP has rate-based flow control which allows it to provide a convenient mechanism for throughput and bandwidth reservation when QoS request is issued.

There are some problems with XTP in regard to supporting continuous media transmission:

1. XTP was designed to be implemented in VLSI to achieve high performance, because it is too complex. However, most current implementations of XTP are done in software and their performance is too slow for transmission of continuous media streams [SGC94].
2. If the round rotation time of the underlying network (e.g., Ethernet) frequently fluctuates, XTP constantly enters the synchronizing handshake which is very undesirable in high-speed networks and for continuous media transmission [AC93].

3. XTP has a *large header*, which creates an overhead of 44 bytes regardless of mode. For example, if an audio stream, with 160-bytes packet size (or less for compressed audio coding) is being transmitted, the header overhead represents 27% of the body content which is more than just a nuisance.

4. Source identification and discrimination are missing in XTP. Source discrimination refers to the necessity to discriminate among several sync and content sources, all arriving through the same network transport association. This feature is important for security and authentication reasons.

5. Internetworking with other protocols is not worked out to provide QoS handling and resource reservation.

Other Transport Protocols

Some other designed transport protocols which are used for multimedia transmission are:

- *Tenet Transport Protocols*

  The Tenet protocol suite for support of multimedia transmission was developed by the Tenet Group at the University of California at Berkeley. The transport protocols in this protocol stack are the *Real-time Message Transport Protocol* (RMTP) and *Continuous Media Transport Protocol* (CMTP). They run above the *Real-Time Internet Protocol* (RTIP).

  The RMTP provides connection-oriented, performance-guaranteed, unreliable delivery of messages. This transport layer is quite lightweight. Two features frequently associated with transport layers, connection management and reliable delivery through retransmission, are absent from this protocol. Thus, the main functions of this transport layer are *flow control* (accomplished by rate control) and the fragmentation and reassembly of messages.

  CMTP is designed to support the transport of periodic network traffic with performance guarantees.

  The RMTP and CMTP provide data and continuous media (periodic network traffic) transmission, but they obey the resource administration done by the

Real-time Channel Administration Protocol (RCAP) which provides resource reservation, admission and QoS handling. Therefore, together the protocol stack also provides guaranteed services with deterministic and statistical QoS bounds [Gup94, BM91].

- *Heidelberg Transport System (HeiTS)*

  The Heidelberg Transport System (HeiTS) is a transport system for multimedia communication. It was developed at the IBM European Networking Center (ENC), Heidelberg. HeiTS provides the raw transport of multimedia over networks. It uses the *Heidelberg Continuous media Realm* (HeiCoRe), which is a real-time environment for handling multimedia data within workstations. The central issue of HeiCoRe and HeiTS together is to provide guaranteed services during multimedia transmission. HeiCoRe includes the *Heidelberg Resource Administration Technique* (HeiRAT), a resource management subsystem that addresses these issues [VHN92].

- *METS: A Multimedia Enhanced Transport Service*

  METS is the multimedia transport service developed at the University of Lancaster [CCH93a]. It runs on top of ATM networks.

  The transport protocol provides an ordered, but non-assured, connection-oriented communication service and features resource allocation based on the user's QoS specification. It allows the user to select upcalls for the notification of corrupt and lost data at the receiver, and also allows the user to re-negotiate QoS levels. The protocol incorporates buffer sharing, rate regulation, scheduling, and basic flow monitoring modules to provide different services, such as guaranteed services with deterministic QoS, statistical QoS bounds and best effort services.

11.2.3 Network Layer

The requirements on the network layer for multimedia transmission are a provision of *high bandwidth, multicasting, resource reservation and QoS guarantees, new routing protocols* with support for streaming capabilities and *new higher-capacity routers* with support of intergated services.

Internet Services and Protocols

Internet is currently going through major changes and extensions to meet the growing need for real-time services from multimedia applications.

There are several protocols which are changing to provide *integrated services*, such as best effort service, real-time service and controlled link sharing. The new service, *controlled link sharing*, is requested by the network operators. They need the ability to control the sharing of bandwidth on a particular link among different traffic classes. They want to be able to divide traffic into few administration classes and assign to each a minimal percentage to the link bandwidth under conditions of overload, while allowing 'unused' bandwidth to be available at other times [BCS93].

- *Internet Protocol (IP)*

  IP provides for the unreliable carriage of datagrams from source host to destination hosts, possibly passing through one or more gateways (routers) and networks in the process. We examine some of the IP properties which are relevant to multimedia transmission requirements:

  - *Type of Service*: IP includes identification of the service quality through the *Type of Service (TOS)* specification. TOS specifies (1) precedence relation and (2) services such as *minimize delay*, *maximize throughput*, *maximize reliability*, *minimize monetary cost* and normal service. Any assertion of TOS can only be used if the network into which an IP packet is injected has a class of service that matches the particular combination of TOS markings selected. The assumption is that different networks offer varying classes of service. Different classes may support different media requirements. For example, multimedia conferencing would need service class which supports low delay, high throughput and intermediate reliability. Precedence handling would support priority schemes in a wide area network and therefore support real-time network traffic. Unfortunately, at present, few commercial routers implement precedence handling in a way that affects the forwarding of packets. In general, no guarantees are provided.

    The TOS capability of IP becomes increasingly important as networks

emerge that have the ability to deliver specific classes of services and offer certain service guarantees. The changes, which may need to be specified, are: for multimedia, we may not need a precedence relation but an *AND* relation; instead of having services, such as *minimize, maximize delay*, etc., lower and upper bounds of a delay should be introduced.

- *Addressing and Multicasting*: One of the most critical functions of the IP is the *addressing*, i.e., to establish a global address space that allows every network in the Internet to be uniquely identified. The IP addressing structure is shown in Figure 11.6. The network addressing structure was

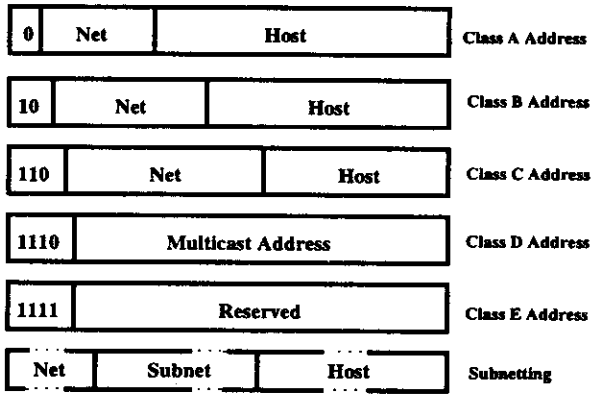

Figure 11.6: *IP addressing structure*

revised to accommodate five classes of address: A, B, C, D and E. *Class A* retained the 24-bit host identifier field, but only seven bits for network number. This address space covers a small number of class A networks. *Class B* with 16 bits for the host identifier and 14 bits for the network number allows a larger number of Class B networks. *Class C* allocates 21 bits for network number and eight bits for host identifier, therefore many more Class C networks are available. A common practice is to assign a class B network address to a collection of LANs.

The LAN technology brought the concept of convenient *broadcasting* to all end-points on the LAN. LANs also introduced the concept of *multicasting*, in which only a subset of the end-points on the LAN are targeted to receive a particular transmission. This capability allows an application

to send a single message to the network and have it delivered to multiple recipients. This service is attractive in a variety of distributed applications, including multi-side audio/video conferencing and distributed database maintenance. This concept is captured in the Internet architecture through special Internet *Class D* addresses, in which multicast-addressed packets are routed (and duplicated at routers when necessary) to all targets that are part of the multicast group.

The realization of multicasting in a WAN environment is a nontrivial undertaking. The routing system must be made aware of which networks have hosts participating in each multicast group, so that the arrival of a multicast-addressed packet can trigger proper forwarding to the destination networks. To avoid duplicative replications of multicast packets by multiple routers, a spanning tree of routers is constructed as part of the multicasting algorithm, to route and duplicate multicast packets.

The worldwide Internet has been providing an IP multicast routing service for some time now, through an Internet segment called *MBone* (Multicast Backbone). The MBone is a collection of UNIX workstations running a routing daemon called "mrouted", which is an implementation of the *Distance Vector Multicast Routing Protocol (DVMRP)* [WPD88]. The MBone is layered on top of the Internet's unicast topology. Connection between the MBone's UNIX workstations is provided by the same Internet links that are used for the usual applications. As multicast datagrams are forwarded from one MBone router to another, they are "tunneled", so that to intervening non-multicast routers they look like regular unicast traffic. Using the MBone, conference sessions from IETF and other Internet technical meetings can be multicast and the remote users can listen to the technical talks and ask the speaker questions.

Class E addresses have been reserved for future extensions.

- *Interconnectivity Between Internet Protocol and Underlying Networks*: The Internet family of protocols is one of today's most widespread protocol stacks in computer networking. There is a strong interest in transporting the IP datagrams, which may carry multimedia traffic, over different networks, for example, an Ethernet or an ATM B-ISDN. Hence, the mapping between the Internet protocol and the underlying layers is

of importance. Another important function in this task is the *binding* of IP addresses to lower-level network addresses. For example, in the case of Ethernet LANs, routers need to encapsulate IP packets in a properly addressed Ethernet packet. Ethernet uses 48-bit addressing. The router learns the binding of the IP to the 48-bit LAN address through the *Address Resolution Protocol* (ARP). This protocol allows a router (or any host) to broadcast a query containing an IP address, and receive back the associated LAN address. A related protocol, *reverse ARP*, can be used to ask which IP address is bound to a given LAN address.

Analogously, in the case of ATM LANs, routers need to encapsulate IP packets in properly addressed cells, and vice versa. Through the ARP protocol, respectively reverse ARP, a mapping between the IP address and VPI/VCI occurs [OG93].

- *Routing*: A major subject in Internet architecture is the *routing* of IP packets because the basic model of Internet consists of networks connected by routers. To create an opportunity for further experimental exploration of different routing protocols for global networking, the concept of *Autonomous Systems (AS)* was developed. ASs are collections of routers falling under a common administrative authority. In theory, the routers commonly use the same routing protocol - *Interior Gateway Protocol (IGP)*, within the AS. AS of gateways (routers) exchange reachability information by means of an *Exterior Gateway Protocol (EGP)*. As the common IGP for the Internet, the *Open Shortest Path First (OSPF)* link-state routing algorithm has been adopted. For EGP , the *Border Gateway Protocol (BGP)* was developed. For routing of OSI connectionless packets the *Inter-Domain Routing Protocol (IDRP)* was standardized in the OSI community.

  For multimedia, the best performance would be achieved if a fixed path (static route) could be allocated, because along this path, guarantees can be met and no or little jitter is experienced. The problem with this extension is that the IP protocol would lose the ability to bypass link-layer failures, which is a fundamental property of the Internet architecture, and should be retained for integrated services. Further, in the case of a static route, if no resource reservation would be performed along the fixed path,

the flexibility of changing a route (in the case of congestion) on a packet basis would be lost, which would decrease the performance of the *best effort* service. Hence, it is very difficult to achieve.

- *Internet Group Management Protocol (IGMP)*

  Internet Group Management protocol (IGMP) is a protocol for managing Internet multicasting groups. It is used by conferencing applications to join and leave particular multicast group. The basic service permits a source to send datagrams to all members of a multicast group. There are no guarantees of the delivery to any or all targets in the group.

  Multicast routers periodically send queries (*Host Membership Query messages*) to refresh their knowledge of memberships present on a particular network. If no reports are received for a particular group after some number of queries, the routers assume the group has no local members, and that they need not forward remotely originated multicasts for that group onto the local network. Otherwise, hosts respond to a query by generating reports (*Host Membership Reports*), reporting each host group to which they belong on the network interface from which the query was received. To avoid an "impulsion" of concurrent reports there are two possibilities: either a host, rather than sending reports immediately, delays for a D-second interval the generation of the report; or, a report is sent with an IP destination address equal to the host group address being reported. This causes other members of the same group on the network to overhear the report and only one report per group is presented on the network.

  Queries are normally sent infrequently, so as to keep the IGMP overhead on host and routers very low. However, when a multicast router starts up, it may issue several queries to quickly build up its knowledge of local membership.

  When a host joins a new group, it should immediately transmit a report for that group, rather than waiting for a query, in case it is the first member of the group.

  IGMP is specified in RFC 1112 and uses an IP packet format with additional IGMP fields such as IGMP type (Host Membership Query, Host Membership Report), checksum, or group address [Dee89].

In a multimedia scenario, IGMP must loosely cooperate with an appropriate resource management protocol, such as RSVP, to provide a resource reservation for a member who wants to join a group during a conference.

- *Resource reSerVation Protocol (RSVP)*

  RSVP is a protocol which transfers reservations and keeps a state at the intermediate nodes. It does not have a data transfer component. RSVP messages are sent as IP datagrams, and the router keeps "soft state", which is refreshed by periodic reservation messages. In the absence of the refresh messages, the routers delete the reservation after a certain timeout.

  This protocol was specified by IETF to provide one of the components for integrated services on the Internet. To implement integrated services, four components need to be implemented: the *packet scheduler*, *admission control routine*, *classifier*, and the *reservation setup protocol*. The RSVP protocol was designed to satisfy requirements, such as [BCS93]:

  - It must accommodate heterogeneous service needs.
  - It must give flexible control over the manner in which reservations can be shared along the branches of the multicast delivery tree.
  - It must accommodate elementary actions such as adding one sender and/or receiver to an existing set, or deleting one.
  - It must be robust enough to scale well to large multicasting groups.
  - It must provide for advance reservation of resources, and for the preemption that this implies.

  A *reservation* specifies the amount of resources to be reserved for all, or some subset of the packets in a particular session. A resource reservation requires adding a *flow* specification control state in the routers which represents an important and fundamental change to the Internet model because the Internet architecture was founded on the concept that all flow-related states should be in the end-systems [BCS93].

  Hence, the resource quantity is specified by the *flow-spec*, which parameterizes the packet scheduling mechanism. The packet subset to receive those

resources is specified by a *filter-spec*. A filter-spec defines a packet filter that is instantiated in the classifier.

The RSVP protocol mechanism provides a very general facility for creating and maintaining a distributed reservation state across the mesh of a multicast delivery path.

RSVP reservations are receiver-oriented, which means that the sender starts, but the actual reservation of resources is performed by the receiver. This is done to support heterogeneous receivers in a multicast group.

STream Protocol, Version 2 (ST-II)

ST-II provides a connection-oriented, guaranteed service for data transport based on the stream model [Top90]. The connections between the sender and several receivers are setup as uni-directional connections, although also duplex connections can be setup.

ST-II is an extension of the original ST protocol [For79]. It consists of two components: the *ST Control Message Protocol (SCMP)*, which is a reliable, connectionless transport for the protocol messages and the *ST* protocol itself, which is an unreliable transport for the data.

ST-II provides a resource reservation during the connection setup. The reservation is originated from the source. It sends a SCMP message with a *flow specification*, which describes the stream requirements (QoS) in terms of packet size, data rate, etc. If the destination accepts the call, it returns the final flow specification to the source. The resource reservation scheme is presented in more detail in Section 11.3.3.

To perform processing in the nodes, ST data packets do not carry complete addressing information. They have a *HOP IDentifier (HIP)*, similar to a virtual circuit number, which is negotiated for each hop during the setup phase.

ST-II is suitable for multimedia transmission because of its resource reservation along the path between the sender and receiver.

Real-Time Internet Protocol (RTIP)

In the Tenet scheme, the services of RTIP (Real-Time Internet Protocol) are used. RTIP provides for connection-oriented, performance-guaranteed, unreliable delivery of packets [VZ91]. It occupies an analogous place in the Tenet protocol stack as the IP in the Internet protocol suite. It communicates with RCAP for resource reservation, therefore it provides guaranteed service [Mah93].

The Tenet protocol suite was designed for real-time communication, with particular emphasis on multimedia transmission.

11.3 Quality of Service and Resource Management

The user/application requirements on the Multimedia Communication System (MCS) are mapped into communication 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 parametrized. 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.

11.3.1 Basic Concepts

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

QoS Layering

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 MCS, the QoS notion must be extended because many other services contribute to the end-to-end service quality. To discuss further QoS and resource management, we need a *layered* model of the MCS with respect to QoS. We assume throughout this section the model shown in Figure 11.7. The MCS 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 in-

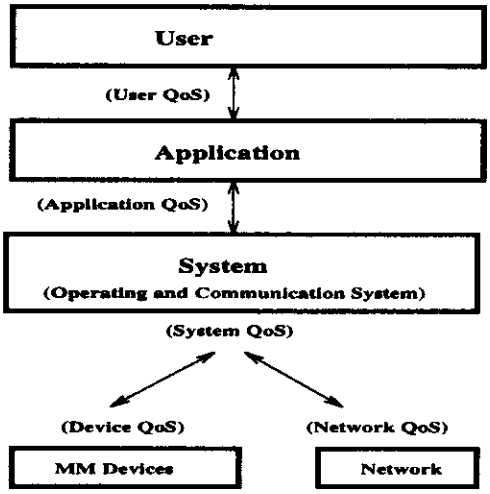

Figure 11.7: *QoS-layered model for the MCS.*

troduction of QoS in the *application* (application QoS), in the *system* (system QoS) and in the *network* (network QoS). In the case of having a human user, the MCS may also have a user QoS specification. We concentrate in the network layer on the network device and its QoS because it is of interest to us in the MCS. The MM devices find their representation (partially) in application QoS.

Service Objects

Services are performed on different objects, for example, media sources, media sinks, connections and 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 Tenet protocol suite, the

services operate over a *"real-time" channel* of a packet switched network [FV90]. In METS, a QoS parameter specification is given for *call, connection and VC* objects [CCH93a]. In RSVP (Resource Reservation Protocol), a flow specification is given for parameterization of the *packet scheduling mechanism in the routers or hosts* [ZDE+93]. At higher layers in communication systems, the service objects, for example *media* [NS95a] or *streams* [SE93], may be specified.

QoS Description

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 MCS.

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 [NS95a].

System QoS parameters describe requirements on the communication services and OS services resulting from the application QoS. They may be specified in terms of both *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 include throughput, delay, response time, rate, data corruption at the system level and 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. The expected services can be associated with specific parameters. For example, the interstream synchronization can be defined through an acceptable skew within the particular data stream [SE93]. Qualitative criteria can be used by the *coordination control* (see Section 11.1.2) to invoke proper services for particular applications.

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 [FV90]; and (2) *network performance*, describing the requirements which the network services must guarantee. Performance might be expressed through a *source-to-destination delay* bound for the connection's packet and packet loss rate [FV90]. Generally, performance bounds are chosen for QoS parameters, such as latency, bandwidth, or delay-jitter, where delay jitter is the maximum difference between end-to-end delays experienced by any two packets [ZK91], but also other parameters for control of QoS (e.g., priority). Note that network services depend on a *traffic model* (arrival of connections requests) and perform according to *traffic parameters*, such as peak data rate or burst length. Hence, calculated traffic parameters are dependent on network QoS parameters and specified in a *traffic contract.*

Device QoS parameters typically specify timing and throughput demands for media data units.

QoS Parameter Values and Types of Service

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

Guaranteed services provide QoS guarantees, as 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., minimum and average value, lowest quality and target quality) or an *interval of values* (lower bound is the minimum value and upper bound is the maximum value). Guaranteed services may also deal with statistical bounds of QoS parameters [FV90], such as statistical bound on error rate etc.

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

[CSZ92].

Best-effort services are services based on either 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.

Note that various systems may provide different classifications of services. An example is the classification of integrated services in the current proposal of the modified Internet architecture [BCS93]. In addition to *best-effort* and *real-time services* (guaranteed service), a service such as *controlled link sharing* is part of the service model (see Section 11.2.2.)

Resource

A *resource* is a system entity required by tasks for manipulating data. Each resource has a set of distinguishing characteristics [Ste94b]:

- 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 or *shared* between various processes. For example, a loudspeaker is an exclusive resource, whereas bandwidth is a shared resource.
- 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 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 MCS can be mapped into three main system resources: *bandwidth of communication channels*, *buffer space* and *CPU processing power*.

Resource Management Architecture

Resources are managed by various components of a resource management subsystem in a networked multimedia system (Figure 11.8). The main goal of resource

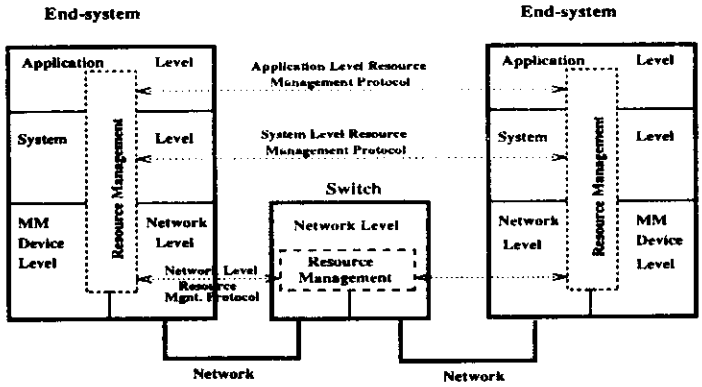

Figure 11.8: *Resource management in MCSs.*

management is to provide guaranteed delivery of multimedia data. This goal implies three main actions: (1) to *reserve* and *allocate* resources (end-to-end) during multimedia call establishment so that traffic can flow according to the QoS specification, which means distribution and negotiation of the QoS specification for system components involved in the data transfer from the source(s) to the sink(s); (2) to *provide* resources according to the QoS specification, which means adhering to resource allocation during multimedia delivery using proper service disciplines; and (3) to *adapt* to resource changes during on-going multimedia data processing.

The resource management subsystem includes *resource managers* at the hosts as well as the network nodes. *Resource management protocols* are used to exchange information about resources among the resource management. The specific tasks of both resource managers and their protocols are discussed in the next subsections.

Relation between QoS and Resources

QoS parameters specify the resource quantity allocated to the services, as well as the service disciplines managing the shared resource in MCS. For example, the end-to-end delay QoS parameter determines the behavior of transmission 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 QoS and resources is embedded in the form of different mappings between QoS parameters and their corresponding resources in 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 their 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, as shown in Figure 11.9, the transmission of video data between a camera, connected to a computer server, and the screen of the computer user involves a resource manager for all depicted components.

11.3.2 Establishment and Closing of the Multimedia Call

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

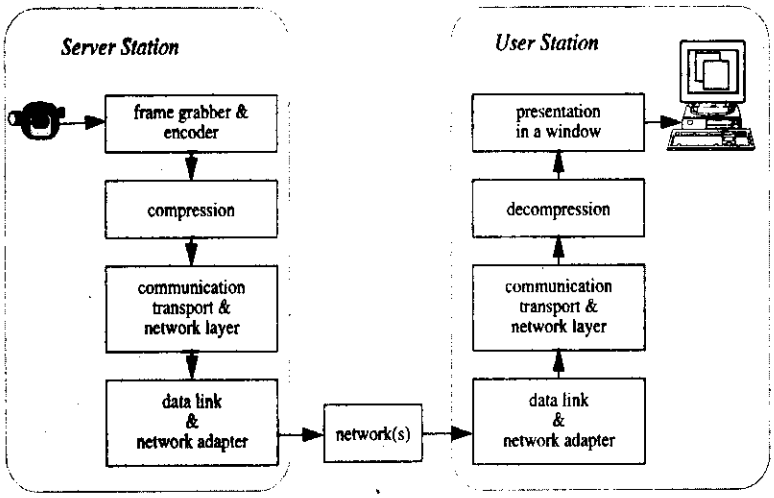

Figure 11.9: *Components grouped for the purpose of video data transmission.*

procedure, from the resource management point of view, concerns resource deallocation.

QoS Negotiation

If we assume that the user has defined the multimedia application requirements, these requirements must be distributed to the resource management entities of all involved system components. The distribution 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, 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 11.10).

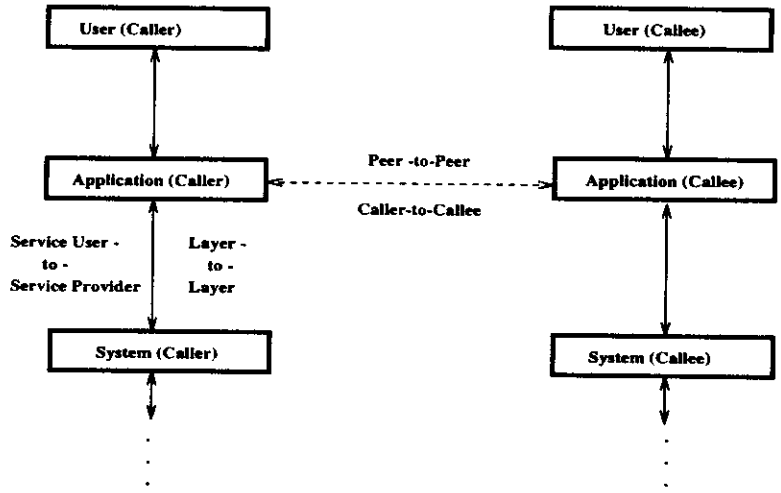

Figure 11.10: *Negotiation.*

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 11.11).

- *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*

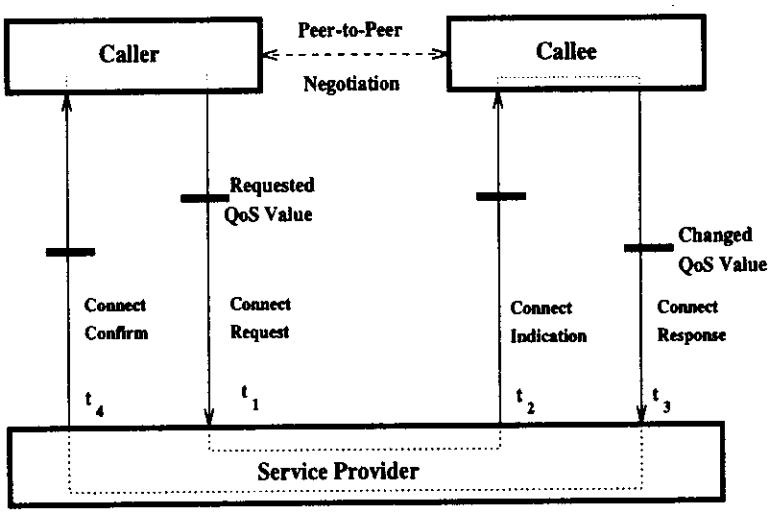

Figure 11.11: *Bilateral peer-to-peer negotiation.*

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" [DBB+93]. 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. A similar case occurs in TV broadcasting. The color TV signal is broadcast uniformly to every user, but users with black and white TVs can still watch the TV program, i.e., the control of the quality is done at the receiver device.

- *Hybrid Negotiation*

  In the case of broadcast/multicast communication, every participating host-receiver may have different capabilities from the 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 unilateral negotiation between network and host-receiver as described above.

- *Triangular Negotiation for Information Exchange*

  In this type of negotiation, the calling user introduces into the request primi-

tive 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 QoS parameter value (Figure 11.12).

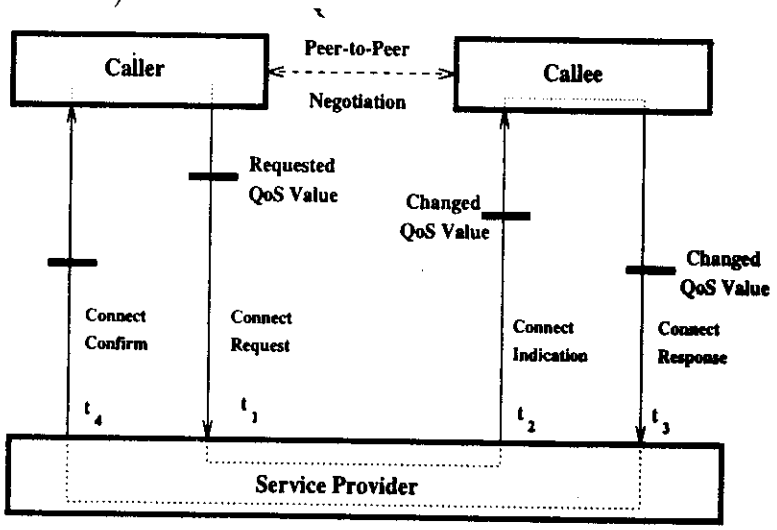

Figure 11.12: *Triangular negotiation for information exchange.*

- *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 it 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 to the caller by the confirm primitive (Figure 11.13) [DBB+93].

- *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

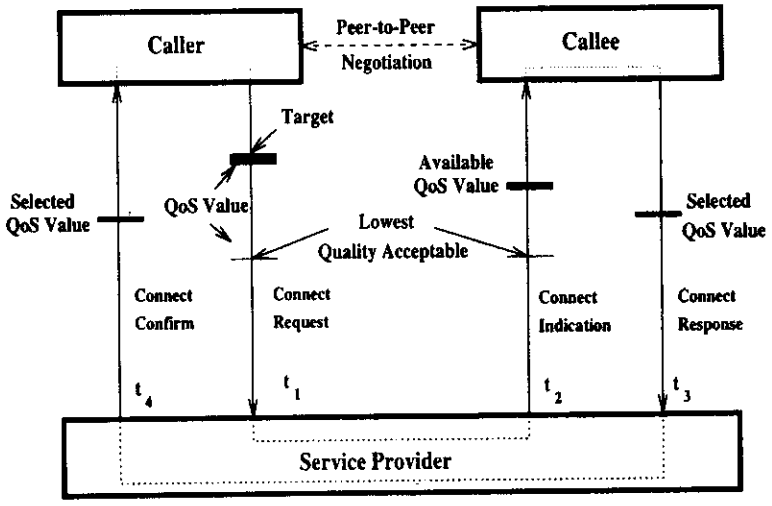

Figure 11.13: *Triangular negotiation for bounded target.*

a contractual value, which in 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 a response/confirm primitive to the caller. The contractual value can also be the maximal QoS parameter value, or threshold [DBB+93] values, which the service user wants to achieve as a contract value.

There are still very few call establishment protocols which have negotiation mechanisms built in. We present some examples which have some notion of negotiation in them.

The *ST-II protocol* provides end-to-end guaranteed service across the Internet network [WH94]. 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* [BM91], *RSVP* [ZDE+93] and others use triangular negotiation for different QoS parameter values. The QoS broker is another end-to-end establishment protocol [NS95a] 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 11.14).

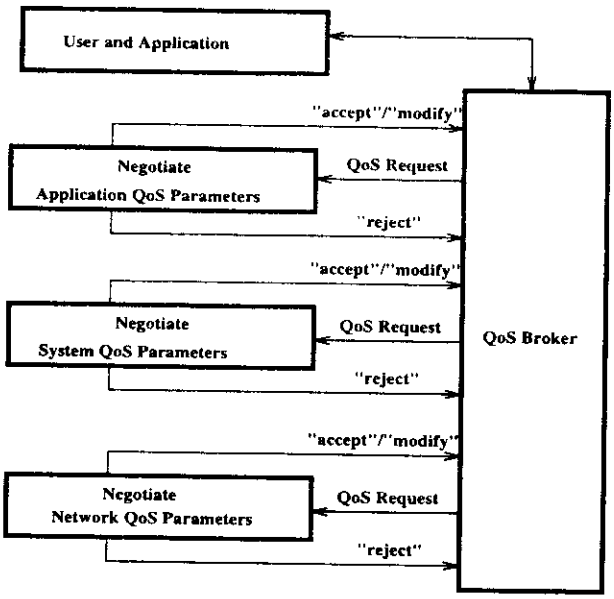

Figure 11.14: *Negotiation in QoS broker.*

Translation

It is widely accepted that different MCS components require different QoS parameters, for example, the mean loss rate, known from packet networks, has no meaning as a QoS video capture device. Likewise, frame quality is of little use to a link layer service provider because the frame quality in terms of number of pixels in both axes

is a QoS value to initialize frame capture buffers.

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 establishment phase. The split of parameters, shown in Figure 11.7, 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 a *tuning service*. A tuning service provides a user with a Graphical User Interface (GUI) 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. Example of such a GUI is shown in Figure 11.15 [NS95b].

- *Application QoS - System QoS*

  Here, the translation must 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 [SE93], or from video frame size to transport packet size. It may also be connected with possible segmentation/reassembly 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 be *bidirectional trans-*

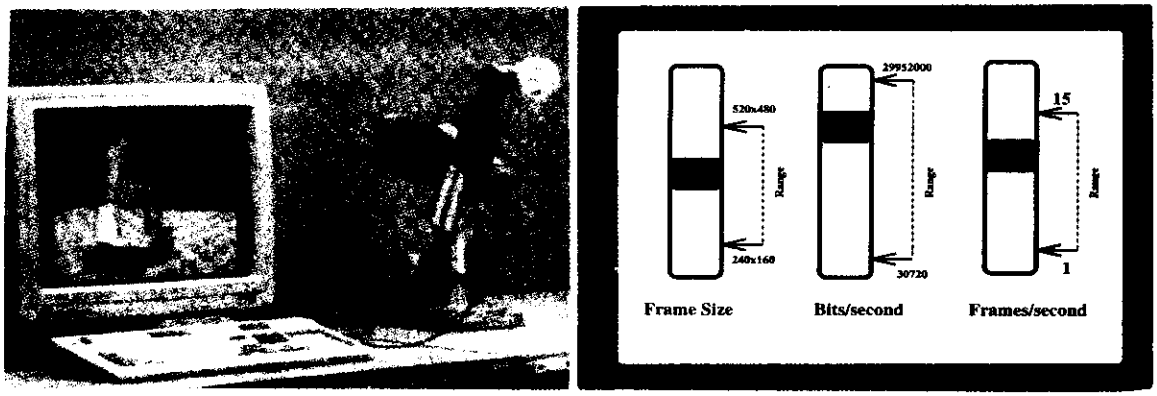

Figure 11.15: *Graphical user interface.*

lation [NS95a]. 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) to reduce the frame size (always keeping the same ratio between horizontal and vertical resolution) until we encounter 112 pixels in the horizontal direction; (2) to reduce the frame rate until we have one frame per second; and, (3) to provide an indication that no further reduction is possible and the connection must be closed.

The reverse translation results in *media scaling.* In general, media scaling methods perform different degrees of media quality degradation if resources are not available. The dynamic QoS change (translation, negotiation, renegotiation of QoS) is used in conjunction with scaling techniques [TTCM92].

Scaling

Scaling means to subsample a data stream and only present a fraction of its original contents. In general, scaling can be done either at the source or at the receiver.

Scaling methods, used in an MCS, can be classified as follows [DHH+93]:

- *Transparent scaling* methods can be applied independently from the upper protocol and application layers. This means that the transport system scales the media down. Transparent scaling is usually achieved by dropping some portions of the data stream. These portions need to be identifiable by the transport system.
- *Non-transparent scaling* methods require an interaction of the transport system with the upper layers. This kind of scaling implies a modification of the media stream before it is presented to the transport layer. Non-transparent scaling typically requires modification of some parameters in the coding algorithms, or even recoding of a stream that was previously encoded in a different format.

In an MCS, scaling can be applied to both audio and video data:

- For *audio*, transparent scaling is difficult because presenting a fraction of the original data is easily noticed by the human listener. Dropping a channel of a stereo stream is an example. Hence, non-transparent scaling must be used for audio streams. For example, a change of the sampling rate at the stream source achieves an audio scaling.
- For *video* streams, the applicability of a specific scaling method depends strongly on the underlying compression technique. There are several domains to which scaling can be applied [DHH+93]:
  - *Temporal scaling* reduces the resolution of the video stream in the time domain. This means that the number of video frames transmitted within a time interval decreases. Temporal scaling is best suited for video streams in which individual frames are self-contained and can be accessed independently.
  - *Spatial scaling* reduces the number of pixels of each image in a video stream. For spatial scaling, hierarchical arrangement is ideal because it has the advantage that the compressed video is immediately available in various resolutions.

- *Frequency scaling* reduces the number of DCT coefficients applied to the compression of an image.
- *Amplitude scaling* reduces the color depth for each image pixel. This can be achieved by introducing a coarser quantization of the DCT coefficients, hence requiring a control of the scaling algorithm over the compression procedure.
- *Color space scaling* reduces the number of entries in the color space. One way to realize color space scaling is to switch from color to gray-scale presentation.

A combination of these scaling methods is possible. In the case of non-transparent scaling, frequency, amplitude and color space scaling are applied at the source during the encoding of the video. Transparent scaling may use temporal scaling or spatial scaling.

Resource Admission

The next step, after every layer inquires or gets its own QoS specification through negotiation and translation, is resource admission. The admission service is an important service at every node along the path between source (sender) and sink (receiver) to check resources for availability. In networks, it is the mechanism used to accept or reject new connections. 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, the reservation protocol creates either a "reserve" message with admitted QoS values or a "reject" message when the minimal bound of QoS values cannot be satisfied. The admitted QoS values may be lower than the target value, but they may still be above the minimal value as shown in Figures 11.12 and 11.13.

There are three types of tests which admission should perform: (1) a *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) a *spatial test* for buffer allocation for delay and reliability guarantees; and, (3) a *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 [HLG93], [Kes92], etc. 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 have prior knowledge of 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, 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.

Resource Reservation/Allocation

For the provision of guaranteed QoS in MCS, *reservation and allocation 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 and allocation of resources in most systems is *simplex*, i.e., the resources are reserved only in one direction on a link, which implies that the senders are logically distinct from receivers.

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, for example, a reservation for the longest processing time of the CPU or the highest bandwidth needed by a task. Resource conflicts are therefore avoided. This leads potentially to an underutilization of resources. This method results in a guaranteed QoS.

- 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 overload and to solve the problem should be implemented. The monitor function then preempts processes according to their importance.

Both approaches represent points in a continuum because the process requires a resource in a stochastic fashion. This requirement has both an average and a peak value. One can assign to it any value between the average and the peak value. The closer the assignment is to the peak value, the lower the probability that the process will be denied the use of the resource at a certain time. Hence, the assignment represents a tradeoff between the peak rate multiplexing (pessimistic approach) and the statistical multiplexing (optimistic approach).

Additional mechanisms to detect and solve resource conflicts must be implemented. The resource managers (e.g., in HeiRAT [WH94]) 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 establishment phase, calculates the QoS guarantees that can be given to the new connection and reserves the corresponding resource capacities.

The reservation and allocation of resources depends on the *reservation model*, its *protocols* and a set of *resource administration functions*, such as admission, allocation, monitoring and deallocation, for individual resources.

- **Reservation Model**

  There are three types of reservation models: (1) *Single Sender/Single Receiver* (e.g., RCAP); (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* [ZDE+93]. 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 [ZDE+93]. 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 [ZDE+93]:

  - *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 N 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*, which is

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 time when 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.

- **Resource Reservation/Allocation 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 applications. Resource reservation protocols are control protocols embedded in a multimedia call establishment protocol. The resource reservation protocol implies that every node and host has 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 with network management agents for proper reservation and administration decisions.

  The resource reservation protocols work generally as follows: the initiator of the connection (e.g., 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). Figure 11.16 shows sender-initiated resource reservation/allocation protocol with "accept" response.

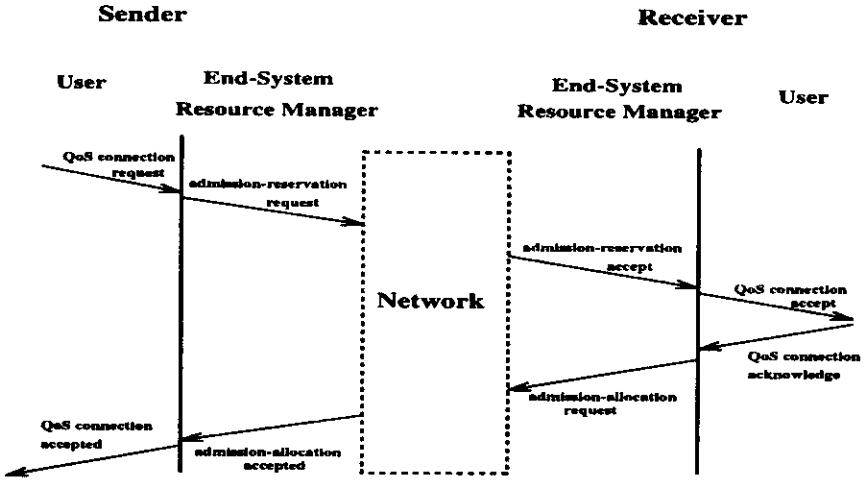

Figure 11.16: *Resource reservation/allocation protocol with "accept" response*

Resource Deallocation

After the transmission of media, resources must be deallocated, which means, the CPU, network bandwidth and buffer space must be freed and the connections through which the media flow must be torn down. The tear down process must 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) must be deallocated and the resource availability must be updated at every node; (2) *Receiver* requests closing of the multimedia call. This request is sent to the sender and during the traversing of the path, the resources are deallocated.

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

11.3.3 Managing Resources during Multimedia Transmission

QoS guarantees must be met in the application, system and network (Figure 11.7) to get the acceptance of the users of MCS. There are several constraints which must be satisfied 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 rate control mechanisms for delay, delay-jitter and throughput (bandwidth) provision, as well as error control for reliability provision. Process and buffer management are presented in Chapter 9 (Operating Systems).

We assume at this point that proper resource reservation and allocation has occurred as described in the previous subsections.

Rate Control

If we assume an MCS to be a tightly coupled system, which has a central process managing all system components, then this central instance can impose a synchronous data handling over all resources; in effect we encounter a fixed, imposed data rate. However, an MCS usually comprises loosely coupled end-systems which communicate over networks. In such a setup, rates must 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 MCS, 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 supports explicit resource allocation and admission control policies.

A *rate-based service discipline* is one that provides a client with a minimum service rate independent of the traffic characteristics of other clients. Such a discipline, operating at a switch, 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 [ZK91]:

- *Fair Queuing*

  If N channels share an output trunk, then each one should get *1/Nth* of the 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 the 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)*. A virtual transmission time is allocated to each packet. 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)*

  Delay EDD [FV90] 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. 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. 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 Stop-and-Go can be achieved using multiple frame sizes, which means it may operate with various frame sizes.

- *Hierarchical Round Robin (HRR)*

  An 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.

Some other disciplines. suitable for providing guaranteed services, are schemes such as the *Weighted Fair Queueing (WFQ)* algorithm [CSZ92]. In WFQ, each packet is stamped with a time-stamp as it arrives and then it is transmitted in increasing order of the time-stamps.

To bound the delays in rate-based disciplines, *traffic shaping* schemes can be applied at the sending host, for example, *Leaky Bucket* and its variations (e.g., *Token Bucket*) [CSZ92]. In Leaky Bucket, the sending host places the data into a bucket and the data drain out the bottom of the bucket, being sent on the network at a certain rate. The rate is enforced by a regulator at the bottom of the bucket. The bucket size limits how much data may build up waiting for the network.

Rate-based disciplines are divided 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 (e.g., Delay EDD, Virtual Clock, Fair Queuing); and (2) the *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 (e.g., Stop-and-Go, HRR, Jitter EDD).

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.

The buffer space requirements for the three non-work-conserving disciplines are almost constant for each node traversed by the channel [ZK91]. 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 (also by Weighted Fair Queueing). Jitter guarantees are provided by Stop-and-Go and Jitter EDD.

End-to-End Error Control

Multimedia extensions to existing operating systems provide a fast and efficient data transport between sources and destinations 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, the 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 MCSs is necessary because of the 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 disk, there are often less stringent real-time requirements for the receiver.

To ensure required reliability of MCSs, end-to-end error control consists of two steps *error detection* and *error correction*.

- *Error Detection*

  Reliability should be enforced, although there is some error tolerance in the multimedia systems. This works 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 only 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, but not always content. This also implies that existing error detection mechanisms, such as *checksumming* and *PDU sequencing*, must be extended toward 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 the decision should actually 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 must be extended too because of the "lateness" concept. It means that if a PDU arrives too late at the receiver, this information is useless for an application and should be detected as an error. 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)*[Mil93a].

- *Error Correction*

  The traditional mechanism for reliability provision is *retransmission* (e.g., TCP protocol uses this mechanism), which uses an acknowledgment principle after receiving data or window-based flow control. If the acknowledgment is negative, the data are 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; and, (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 MCS currently discussed in the 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 sends a negative acknowledgment if PDU i is lost. The problem is that if after that i'th PDU the packets were transmitted successfully, they are dropped too, which may lead to several implications: (1) *gap introduction* (Figure 11.17); 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 must 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 a mechanism such as freezing the video, or turning down the audio.

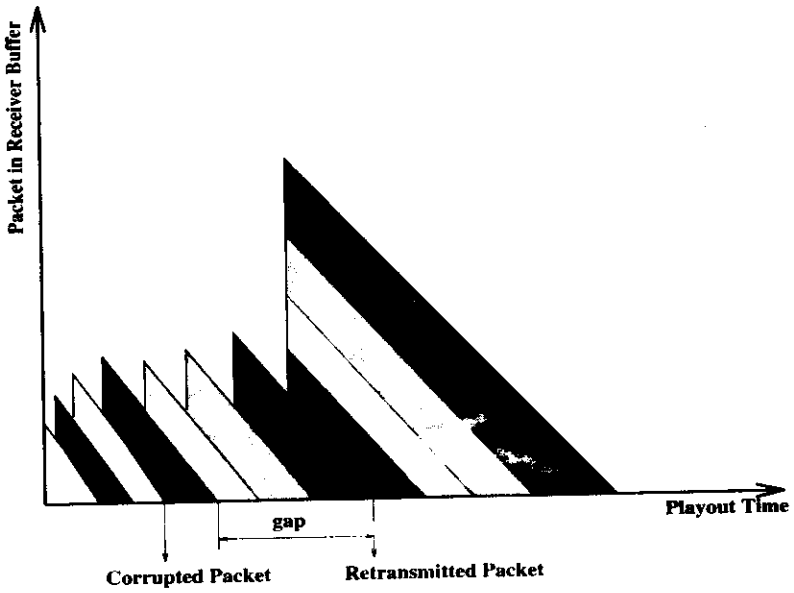

Figure 11.17: *Gaps in Go-back-N retransmission.*

– *Selective Retransmission*

Selective retransmission provides better channel utilization. The receiver sends a negative acknowledgment 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 must 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.

- *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 setup between the sender and 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 [Bie93]. 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 pieces of informations: (1) the error probability of the network between the sender and receiver; and (2) the 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 [Bie93].

Also, FEC requires hardware support at end-systems 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 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 observes *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 to buffer sufficient packets to provide for continuous playback in the presence of jitter. The voice data consist of talk spurts and periods of silence. Since talk spurts are generally isolated from each other by relatively long silence periods, voice protocols typically impose the control time on the first packet of each talk spurt. 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 to achieve a zero-gap playback schedule

for the talk spurt. 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 playback. In the latter 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 talk spurt and to use it so that the slack time of arriving packets is lengthened. An example is shown in Figure 11.18 [DLW93].

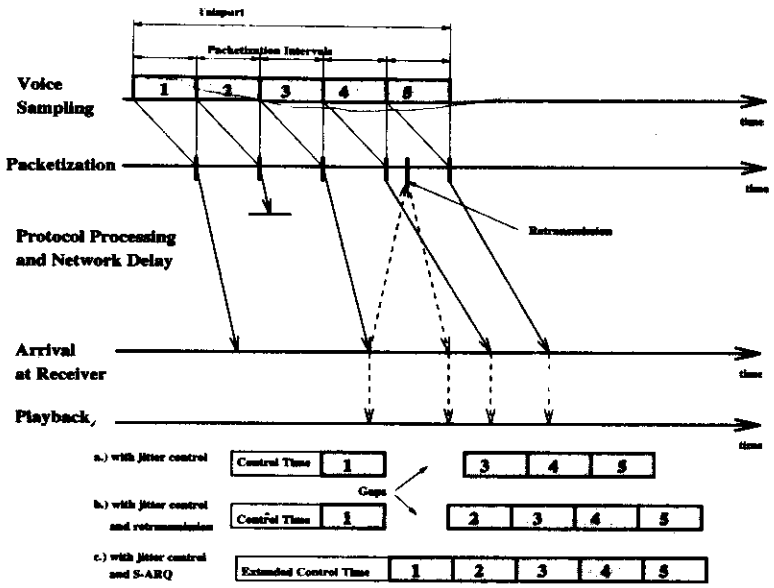

Figure 11.18: *S-ARQ*.

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 of working properly if we introduce large end-to-end delays with large buffers. Hence, preventive schemes should be used.

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. 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). The Management Information Base (MIB) may also be used for resource admission. To use network management for resource management, the MIBs of the network management must 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 a 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 [WH94]. 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 latter reports regularly the resource status on different nodes along the path between communicating end-users.

Monitoring at end-systems includes a *supervisor function* to continuously observe that 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 Mbits/s over a duration of at most three frame lengths. However, at some point in time the system starts to deliver a continuous bit rate of 6 Mbits/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.

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.* In 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 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 the Tenet protocol stack.

Another classification of administration protocols can be done according to the criteria: is the administration protocol part of the transmission protocol, or not. An example for coupling the establishment, transmission and management phases is the ST-II protocol. ST-II is further coupled to SNMP-MIB for management. An example where the administration protocol is decoupled from transmission protocols, is RCAP [BM91]. 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.

Resource Adaptation

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 MCS architecture 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 must be provided 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.

During multimedia transmission, change of QoS parameters and 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 setup. The renegotiation request can come either from the user, who wants to change the quality of service, 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 the resource manager. A possible signaling paradigm for renegotiation at the end-point is shown in Figure 11.19.

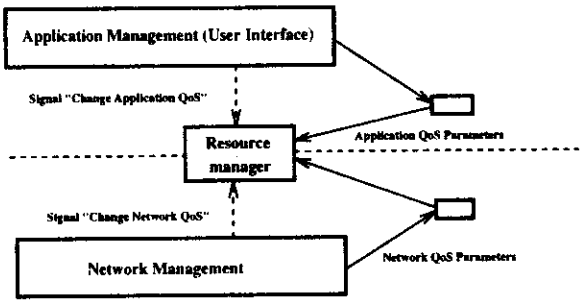

Figure 11.19: *Signaling paradigm for renegotiation at the end-point.*

- *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 resource manager must 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 must be invoked to check the availability of network resources if the change of QoS requires change of network resources. If resources are available, resource reservation and allocation is performed.

  If the user-receiver requires change of QoS for the receiving media, first the resource manager checks the local resource and reserves it. Then, the sender is notified via a 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 must 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.

- *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 users) violate their admitted application requirements. Then, a notification about the degradation of the QoS performance and a 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 resource manager may invoke the resource administration protocol to lower the QoS parameters in the network between the sender and receiver.

- *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 must 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 must react to this change. In the latter case, the source (host) must adapt.

We describe several 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 the host system.

- *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 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 resource allocation, the MORE guaranteed connection requests are 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) and *load balancing control* (making a decision to re-route a channel) [PZF92]. 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 it 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 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* [PZF92] of 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.

- *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 [KMR93], 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 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.

  - *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.

- *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 by smoothing over an interval of 1-4 frames [KMR93].

- *Hierarchical Coding*

  Hierarchical coding describes algorithms which produce two or more types of cells describing the same block of pixels with different degrees of detail. However, these coders are more complex and use a greater amount of bandwidth to transmit images than single-layer coders [KMR93].

11.3.4 Architectural Issues

Networked multimedia systems work in *connection-oriented mode*, although the Internet is an example of a connectionless network where QoS is introduced on a packet basis (every IP packet carries type of service parameters because the Internet does not have a service notion). MCS, based on that 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 and connected resource admission, reservation, allocation and provision must 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 must be provided.

The *end-point architectures* 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, must 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.

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 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 to have a global view on network resources.

Some examples of architectural choices where QoS and resource management are designed and implemented include the following:

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 [DBB+93].

2. Lancaster's QoS-Architecture (QoS-A) [CCH93a] offers a framework to specify and implement the required performance properties of multimedia applications over high-performance ATM-based networks. 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) [WH94], 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 and CMTP provides network QoS negotiation, reservation and resource administration through the RCAP control and management protocol.

5. The Internet protocol stack, based on IP protocol, provides resource reserva-

tion if the RSVP control protocol [ZDE+93] is used.

6. QoS handling and management is provided in UPenn's end-point architecture (OMEGA Architecture) at the application and transport subsystems [NS95a], 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.

7. The *Native-Mode ATM Protocol Stack* [KS95], developed in the IDLInet (IIT Delhi Low-cost Integrated Network) testbed at the Indian Institute of technology provides network QoS guarantees.

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.

11.4 Comments

11.4.1 Trends in Collaborative Computing

New application disciplines place new demands on the collaboration infrastructure and new tele-services are emerging. Only conferencing and application sharing is not enough. Multimedia networked applications, such as tele-medicine, tele-working, virtual collaborative space, distributed simulations and tele-action require sophisticated handling at the level of an application subsystem. We discuss these application more in detail in Chapter 17.

Future collaborative computing will incorporate a (possibly unknown) number of people at geographically distributed sites, using a variety of applications from different application domains. With the *heterogeneity* of collaborative multimedia applications, interoperability issues need to be satisfied.

Simplification of Groupware Development

The development process for groupware needs to be simplified through re-usable components. Groupware should have a plug-and-play architecture [SW94a]. Also, collaborative development support during software engineering should be at much higher level.

Shared Abstractions and Standard Interfaces

Shared abstractions and standard interfaces need to be developed to accommodate heterogeneity. The *Rendezvous* architecture from Bellcore provides already a partial solution for graphical shared abstractions [HBP+93]. The Rendezvous architecture is based on a centralized *abstraction* that collects information common to all users into a single space. Each user has his/her own *personal view* to this information. There is a link between each view and the abstraction. The links are responsible for maintaining consistency between each view and the abstraction. Figure 11.20 shows the shared abstraction of different personal views on a graph.

Standard interfaces are strongly supported by the industry. In March 1993, the Hewlett-Packard Company, IBM Corp., SCO (Santa Cruz Operation, Inc.), Sun Microsystems, Inc., Univel and UNIX System Laboratories, Inc. announced their intent to deliver a common open software environment across their UNIX system platforms. They have defined a specification for a common desktop environment that gives end-users a consistent look and feel. They have defined a consistent set of Application Programming Interfaces (APIs) for the desktop that will run across all of their systems, opening up a larger opportunity for software developers [UNI93].

Standard Protocols

Standard protocols will need to incorporate communication mechanisms which support collaborative computing, such as *multicast* or *protocol abstractions*.

Multicast is needed to provide efficient multiway communication (1-to-N, N-to-1, N-to-N). Protocol abstractions, such as *distributed session control*, are needed to

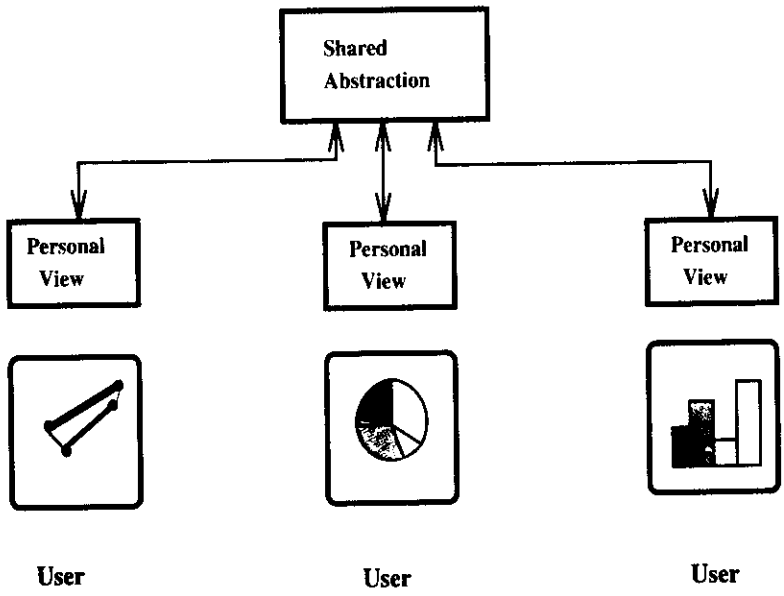

Figure 11.20: *Shared abstraction.*

shield users from the complexity of multipart multimedia coordination. Several abstractions have emerged to describe streams, connections and other objects in communication systems. In the Internet draft [BCS93], the *flow* abstraction was specified to describe the object which is processed by the *integrated services* in the Internet architecture.

Self-describing Media Agents

Descriptive languages are required to characterize varied groupware capabilities and requirements.

Translations

Translations among different CSCW sites are needed to overcome hardware and software heterogeneity, for example, to bridge different media encoding schemes. *Combination nodes* [Luk94], [Sch93] have been proposed to provide translation at

the conjunction between sources and sinks. Their functionality is to *mix*, *compose* an *assemble* the interesting pieces of several video flows into a single flow, where the *selection* may occur either by a sender (chairperson) or receiver (individually tailored). Further, the combination nodes need *translation* functionality between encodings, *reduction* when scalable coding is used and a combination of these operations.

11.4.2 Trends in Transport Systems

The trend in transport systems goes in two directions: one is the *special-purpose protocol approach*, the other one is the *general-purpose protocol approach* [Cha93].

The *special-purpose protocol approach*, also known as the *Internet paradigm*, is to design various *special-purpose* protocols on top of IP for different classes of applications. An example is TCP as a special-purpose protocol for reliable data communication, UDP for unreliable data communication, and RTP for audio and video transport.

The *general-purpose protocol approach* is to provide a general set of services that the user can pick and use. An example is XTP, where the user can select one-way, two-way or three-way handshaking for connection setup and release, etc.

It is hard to predict which paradigm is the right one for the future. More and more special application types are coming. Whether to keep designing and supporting growing protocols in a machine, or use a general protocol to let user pick is a question to answer.

A more realistic and flexible approach may be to develop *application-tailored* protocols that are customized for specific types of services, such as transferring voice, video, text and image data [SP90], [SSS+93], [Nah93]. In our opinion, the transport subsystems move more and more towards a provision of several service classes. Therefore, a transport system may offer, for example, a class of guaranteed services and a class of best-effort services. Within each service class, the services are customized through QoS specification, and accordingly, services inside of the class are selected and customized toward applications.