J. Indian Inst. Sci., May–June 2000, **80**, 195–213.

A design of QoS adaptation and mixer algorithms for wireless multimedia communications

P. VENKATARAM* AND P. SURESH BABU**
Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India.
Email: pallapa, suresh@ece.iisc.ernet.in, Tel: (0091)(80) 360 0855; Fax: (0091)(80) 360 7991

Abstract

The convergence of wireless systems with multimedia services is the challenge of future-generation wireless network. Quality of Service (QoS) provision to support distributed, real-time multimedia applications for mobile users has emerged as an area of major technological focus.

In this work, we propose new adaptation procedures to provide desired QoS in wireless networks which allow the system to recover *automatically*, if possible from QoS violations by identifying a new configuration of system units that might support the initially agreed QoS and by performing a user transparent transition from the original configuration to the new one. We also propose mixer algorithms that can be used in bad atmospheric conditions during heavy data loss due to wireless problems. The proposed approaches, together with suitable negotiation mechanisms, allow us to: (1) reduce the probability of QoS violations which may be noticed by the user, and thus increase user confidence in the service provider, (2) improve the utilization of the system resources, and thus increase the system availability, and (3) deliver the data in time in urgent situations.

1. Introduction

A multimedia environment presently consists of applications accessing pagers, facsimile, answering machines, telephone lines, speech synthesis, and digital recording and playback. Its key contributions are the integration of multiple media into a cohesive nomadic information infrastructure and graceful transition from desktop to nomadic locals. This integration is at the service and user interface levels. It is important to develop new and highly integrated nomadic computing platforms capable of employing emerging digital wireless communications networks.

Multimedia service requirements

A typical distributed multimedia environment in which multiple remotely located users participate in a joint work or design project, demand:[1]

1. *Resource sharing* to integrate information on a more global basis, preserve investments, and ensure the use of a system's available information;

2. *Multimedia data integration* such as images, graphics, sound, text, and structured data, to present information in a more immediate and understandable form;

*Author for correspondence.
**The author is presently with Siemens India Ltd, Bangalore.

3. *Local intelligence and autonomy* to perform tasks independently, in spite of centralized systems;

4. *Graphical interfaces* to reduce training costs and assist occasional and inexperienced users;

5. *Vendor independence* to achieve free from any specific hardware vendor.

1.1. *Problems in wireless communications*

Problems in wireless networks are mainly due to mobility and unreliable nature of the wireless link.[2] Reliable protocols, such as TCP, use end-to-end flow, congestion and error-control mechanisms to provide reliable delivery over an internetwork. However, co-existence of wireless links and mobile hosts with fixed network poses unique problems for transport protocols. In particular, the following communication characteristics of wireless links have significant implications.

1. Maximum transmission unit (MTU) on wireless link is typically much smaller than that over links in the wired networks.[3, 4] Small MTU over the first link forces transmission of smaller packets over the entire end-to-end path through wired path which can accommodate much larger packets.
2. The error rates on wireless link are much higher than those experienced over the links in the wired network.[5–7] Higher error rates and resulting intermittent connectivity over a wireless link are due to a combination of factors such as multipath fading, terrain, and environmental factors, and interference from other transmission. In addition, these errors often cause a burst of packets to be lost.
3. Communications pause during hand-offs are also perceived as periods of heavy losses by transport and higher-level protocols.[8]

These wireless transmission characteristics together contribute to severe degradation in performance of protocols, such as TCP. Use of small packets leads to underutilization of available bandwidth in the wired network and reduces overall end-to-end throughput. Higher error rates and communication pause during hand-off can falsely trigger congestion control mechanism of TCP.[9]

1.2. *Proposed algorithms for quality of service and mixers*

Due to increasing demands of distributed multimedia (DMM) applications, efficient and effective support of quality of service (QoS) has become increasingly important.[10] To support QoS requirements, communication systems and end-systems must provide latency and bandwidth characteristics that allow timely transmission of information. A number of schemes have been proposed to provide deterministic and/or statistical QoS guarantees spanning end-systems and networks. Regardless of the type of guarantees, mechanisms for QoS adaptation are required to deal with QoS violations since temporary overload conditions will be common in future systems.

The proposed QoS-Manager adaptation has been guided by the following premises.

1. QoS adaptation should be performed automatically whenever it is possible.
2. QoS violations should be dealt with locally at the unit level; and
3. QoS adaptation should maintain initially agreed QoS as long as possible; before any QoS degradation is initiated.

We also propose a Mixer algorithm which acts as an intermediate system that receives packets from one or more sources and changes the data format in some manner and then forwards it as a new packet.

2. QoS requirements for multimedia applications

The notion of quality of service originally emerged in communications to describe certain technical characteristics of data transmission. For example, the open systems interconnection (OSI) reference model and TCP/IP model have a number of QoS parameters describing the speed and reliability of transmission, such as throughput, transit delay, error rate, and connection establishment failure probability, etc.[11–13] These parameters apply mostly to lower protocol layers and are not meant to be directly observable or verifiable by the application. Consequently, OSI's QoS coverage is incomplete and even inconsistent.

Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application.[14]

Functionality includes both the presentational multimedia data to the user and general user satisfaction. The QoS of a given system is expressed as a set of (parameter-value) pairs, something called a tuple. We consider each parameter as a typed variable whose value can be a range over the given set. Different applications on the same distributed systems can have different subsets of relevant QoS parameters, with different values required and some parameters might not be mutually independent. In a distributed multimedia system, it is hard to separate the QoS parameters from other system parameters. However, one distinguishing feature is that QoS parameters are subject to negotiation between system units.[15]

2.1. *QoS processing*

Processing of QoS in a distributed multimedia system involves several related activities. Some of them are:

1. Assessing the QoS requirements in terms of users subjective wishes or satisfaction with the quality of the application—performance, synchronization, cost, and so forth.
2. Mapping the assessment results onto QoS parameters for various system units or layers. For example, a user chooses video in terms of its resolution and frame rate, which map onto throughput requirements.
3. Negotiating between system units or layers (embedded in protocols) to ensure that all system units can meet the required parameters consistently.

If the negotiation ends with an agreement on the required values, the application can be launched. Types of agreements include guaranteed, best effort, or stochastic.

3. QoS adaptation algorithms

Due to increasing demands of distributed multimedia (DMM) applications, efficient and effective support of QoS has become increasingly important. To support QoS requirements, the communication systems and the end systems must provide latency and bandwidth characteristics that allow timely transmission of information.[16] DMM applications require a system that *maintains* the initially agreed QoS, *regardless of the fluctuation* in system load; otherwise, DMM applications will not be able to compete against traditional systems, such as television. A number of schemes have been proposed to provide deterministic and/or statistical QoS guarantees for spanning the resource utilization of the end systems and networks. Internet traditionally provides a 'best effort' service without QoS guarantees. DMM applications which depend on a certain level of QoS need a mechanism for QoS adaptation in order to deal with temporary changes in the available QoS parameters. We note that a renegotiation of the QoS parameter values of a DMM application may also be initiated by the user during an ongoing session. The user may wish to increase the quality or reduce it in order to reduce costs. The internal mechanism for adapting the application to this new situation is similar in this case to the mechanism used for adapting to changing network and server QoS parameters.

3.1. *Existing approaches for QoS adaptation*

Two approaches has been proposed for QoS adaptation: *network configuration level and system unit level.*[17]

The first approach involves a reconfiguration of the application infrastructure, replacing the overloaded system units (or simply units) by other alternative units. The second approach does not change the configuration of the system units, but changes the QoS characteristics allocated to different units.

3.1.1. *Adaptation at the network configuration level*

In this approach, when a violation is detected, one or more alternative units are selected, and transparent transition from the primary units to the alternative ones is performed. The alternative units are selected based on factors such as functional configuration of the requested service, and the current load of system units. The QoS characteristics considered by this approach are delay, jitter, throughput and reliability (expressed in terms of loss rate); that is, the approach may be used to recover from delay, throughput, and/or loss rate violations. This may be useful for any application that requires certain guarantees on QoS, such as video-on-demand and teleconferencing systems.

3.1.2. *Adaptation at the system unit level*

In this approach, in order to provide end-to-end QoS guarantees, each system unit involved in the QoS provision must contribute its share to the requested end-to-end QoS.[18] When a unit violates its guarantees (and this is detected by the system), some form of cooperation among the units is started with the aim of reassigning the guarantees to the different units, so that the end-to-end guarantees, as observed by the user are unaffected. Thus, the non-overloaded units may reserve additional resources, e.g. buffers or CPU slots in order to provide an improved

QoS and thus compensate the violations of the other units. The QoS characteristics considered by this approach are delay, jitter and reliability (expressed in terms of loss rate); that is, the approach may be used to adapt from delay, jitter, and loss rate violations. This may be useful for any application with stringent temporal requirements, such as teleconferencing systems.

When the system cannot recover from QoS violations (using either of the approaches), users or applications should be required to intervene. The user should be informed directly at the user interface. If a violation occurs, a special notification is sent to the application/user, who can react accordingly. Generally, the interactions with the user lead to the renegotiation of a degraded QoS or simply to the abortion of the application (as in most existing systems).

3.2. *Schemes for solving QoS adaptation approaches*

We have proposed three schemes to solve the QoS adaptation problem. The first scheme, called *unit reconfiguration scheme* (URS), performs adaptation at the configuration level. The second scheme, called *network resource reconfiguration scheme* (NRRS), recovers from QoS violations by changing the distribution of QoS levels that each system unit that will support in the near future. The third scheme, called *delay recovery scheme* (DRS), recovers from transit delay violations immediately, so that the failure of one or more units to meet their commitment does not necessarily lead to end-to-end QoS violation.

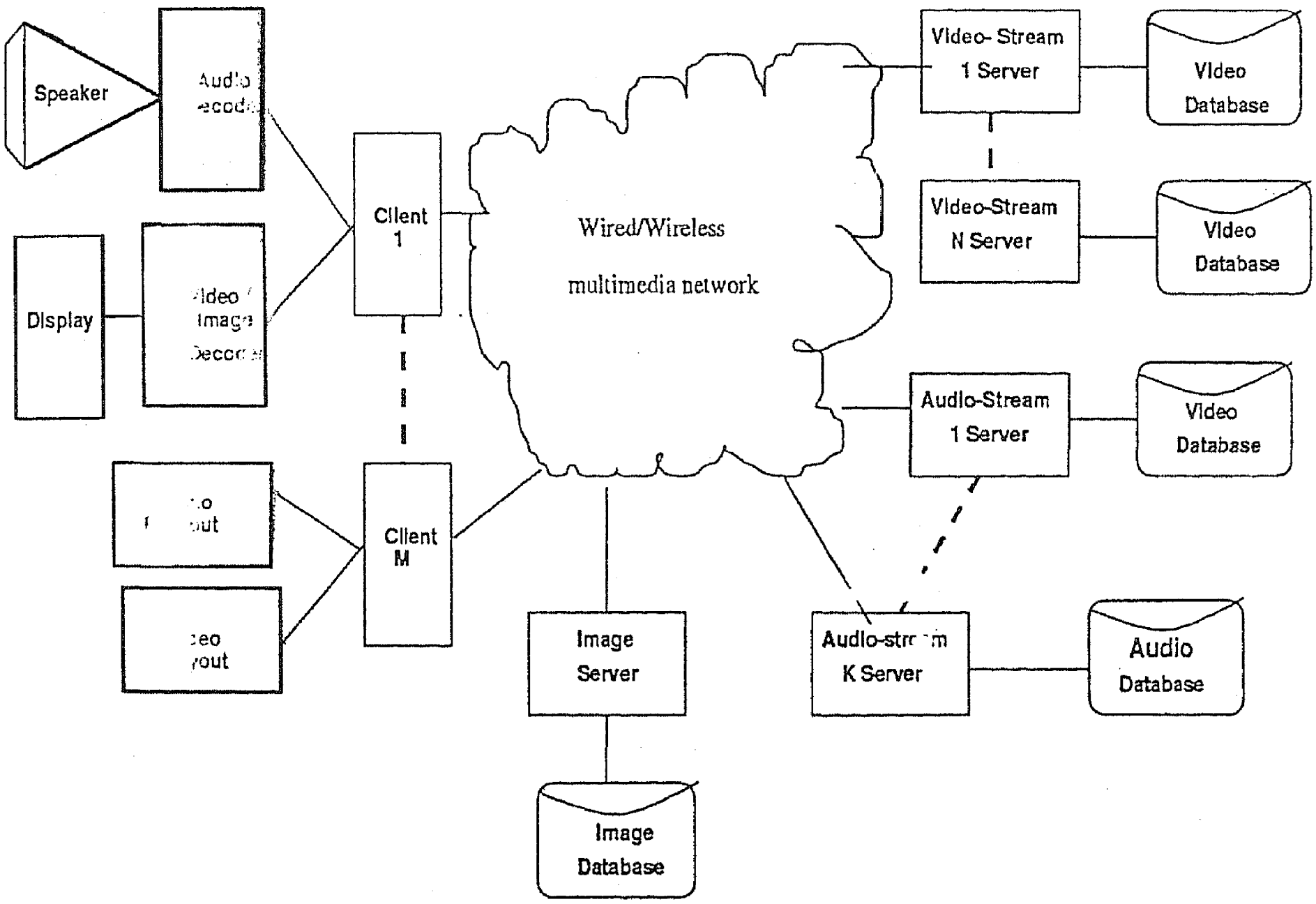

FIG. 1. A distributed multimedia system on a wired/wireless network.

The proposed schemes, together with suitable negotiation mechanisms will make more efficient use of system resources, thus increasing the system availability, and will increase the user acceptance of DMM by decreasing the probability of QoS violations noticed by the users.

For many applications, especially presentational ones, a user views the information coming from a remote multimedia database, on a linear configuration. However, certain applications, such as teleconferencing, require much more complex system architecture due to the large number of users and servers that could be involved (see Fig. 1).

A QoS management framework is assumed where each system unit has its own QoS-agent, which provides means for handling all the information about the performance and functional behavior of the given system unit. QoS adaptation is performed by a distributed QoS-manager which resides on end-systems and gathers information from QoS-agents of intermediate systems and QoS-managers of end-systems.

QoS parameters

The system units, e.g. a network or a decoder, can be characterized by the following QoS parameters, to be evaluated over a certain measurement interval:

- *transit delay* is the time between the moment some data unit is received (at the input port) to the moment it is sent (at the output port);
- *transit delay* jitter indicates the variation of the delays experienced by different data units in the same stream;
- *loss rate* is the fraction of data units lost during transit.

Concatenation properties

If several units are composed in a linear configuration, as shown in Fig. 1, the end-to-end QoS characteristics of such a composition can be calculated based on the QoS characteristics of individual unit.[19, 20] For instance, the delay of the composed system consisting of a network and a decoder is the sum of the network delays and the delay of the decoder processing. In general, the end-to-end QoS parameters of n stream-processing units can be calculated as follows.

$Throughput = \min(\text{for all } i = 1,\ldots, n) \text{ of } Throughput^i$

$$Delay = \sum_{i=1}^{n} Delay^i,$$

$$Log\left(1 - Lossrate^i\right) = \sum_{i=1}^{n} Log\left(1 - Lossrate^i\right)$$

assuming that jitter is defined as the difference between the maximum and minimum delay).

$$Jitter = \sqrt{\sum_{i=1}^{n} \left(jitter^{i}\right)^{2}}$$

(assuming that the jitter is defined as the average deviation of the delay from the average delay, and the delay is assumed to have normal distribution).

These formulae apply to the actual QoS parameters that describe the performance of the system, as observed during its operation. For system management purposes, we are not only interested in these actual QoS parameters, but also in QoS guarantees that may be obtained from the different system units during planning and initialization of the multimedia application. The QOS guarantees constitute the following:

1. deterministic guarantee (which means that the communication service is equal to or better than the specified QoS parameters).
2. statistical guarantee (which means deterministic guarantee for at least a certain fraction, e.g. 95% of the transmitted data blocks would be delivered, or for a certain fraction of the connections that are established over a long period).
3. target objectives (which means that the unit knows the requirements and tries to satisfy them without providing any guarantee), and
4. best effort (which means that the unit will do as well as it can without considering the particular QoS requirements); past experience may provide some information about how well the unit usually performs.

The above formulae remain valid for QoS guarantees as long as all units provide guarantees of the same degree. If this is not the case, the degree of guarantee of the end-to-end QoS parameters is the lowest guarantee provided among the different units. For instance, if one network in the configuration only provides a best effort service, the resulting end-to-end service will only be of the same type.

3.2.1. *The unit reconfiguration scheme*

The unit reconfiguration scheme (URS) leads to the selection of optimal configuration. Figure 2 shows the broad view of the proposed URS procedure. Consider that the QoS-manager of a multimedia application is running at the server and the QoS agent is running at the client. The procedure is as follows:

URS algorithm

Step1: The QoS-manager identifies the units that may get involved in providing the requested QoS to the application. The identification is based on the functional behavior of the application, and the static characteristics of the system units, such as the software functions they support and their maximum capacity.

Step2: The QoS-manager orders the configuration produced in Step1 according to the optimization criteria. The following static information may be used for this purpose: (1) the cost to be

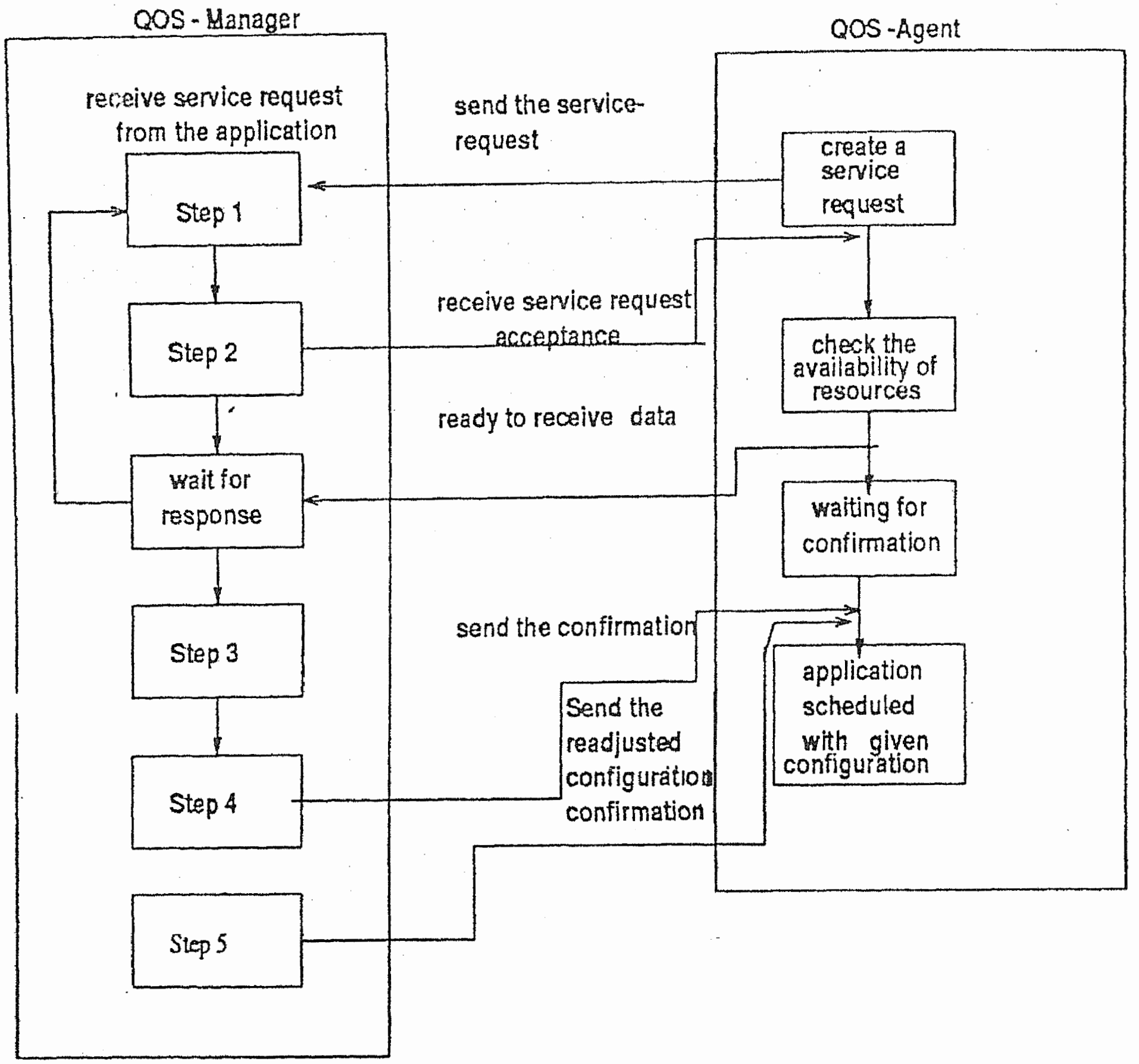

FIG. 2. Broad view of unit reconfiguration scheme (URS).

charged to the user for using a given configuration; and (2) the availability and reliability (statistical values) of the units of a given configuration, and (3) the QoS that might be provided by a given configuration. Several algorithms may be used to classify the configurations depending on the type of applications considered[21];

Step3: The QoS-manager selects the best configuration and inquires about the available service quality from each of the units via their QoS-agents. On receipt of the service request, each QoS-agent makes a resource reservation for the best possible level of QoS and sends this information to the QoS-manager.

Step4: When receiving all responses, the QoS-manager determines whether the configuration meets the end-to-end requirements using the above formulae. If the QoS of the units committing to support does not satisfy the requirements, another configuration is considered. This proceeds until a configuration supporting the requested QoS is found or all configuration identified in Step1 have been checked. If the QoS of the units committed to support is better than the requested QoS, the commitment of certain units can be relaxed.[22]

Step5: The QoS-manager sends a message to each of the QoS-agents in the configuration in order to request the effective reservation of the resources or the deallocation of the resources that have been temporarily reserved.

Responsiveness

The response time of the URS is the time between the moment a QoS violation is detected and the moment reconfiguration has been completed, either resulting in a new configuration that reestablishes the originally agreed QoS or leading to the conclusion that the agreed QoS cannot be maintained. In case the reconfiguration is requested by the user, this results in delays in the response time of the system. However, in case the system automatically detects any internal QoS violation of a unit and invokes the URS before the user notices any end-to-end QoS violation, we also have to take into account the time required by the QoS monitoring tool for diagnosing the QoS violation.

3.2.2. *Network resource reconfiguration scheme*

NRRS is an adaptation scheme proposed to maintain the delay, jitter, and/or loss rate when one or more units of the configuration of interest failed to meet their commitment.

The idea for NRRS is to change, in response to a QoS violation, the amount of resources reserved by the units in the configuration in such a way that the end-to-end requirements will still be met. When a QoS violation occurs, the QoS agent of the overloaded unit (which does not meet the initially agreed QoS) will ask the other units to reserve additional resources to compensate for the violation. This is done through the sending of the so-called *QOS-Violation-indication* (QUI). If the system units do not have enough resources to recover from the violation, a renegotiation is initiated with the applications/users.

The new resource configuration (the new distribution of QoS levels over the units of the configuration) remains in place until:

1. another QoS violation occurs, or /and
2. the overloaded unit recovers from the problem; if the overloaded unit recovers from the problem, it may resume the initially agreed QoS and send a QOS reduction message to the other units to reduce extra commitments.

The responsiveness of NRRS is determined by the monitoring delay and response time of the NRRS algorithm. The response time of the algorithm depends on the type of architecture used—centralized, ring or hierarchical.

Example: Consider an internetwork-based distributed system, as shown in Fig. 3, which uses three networks and a gateway.

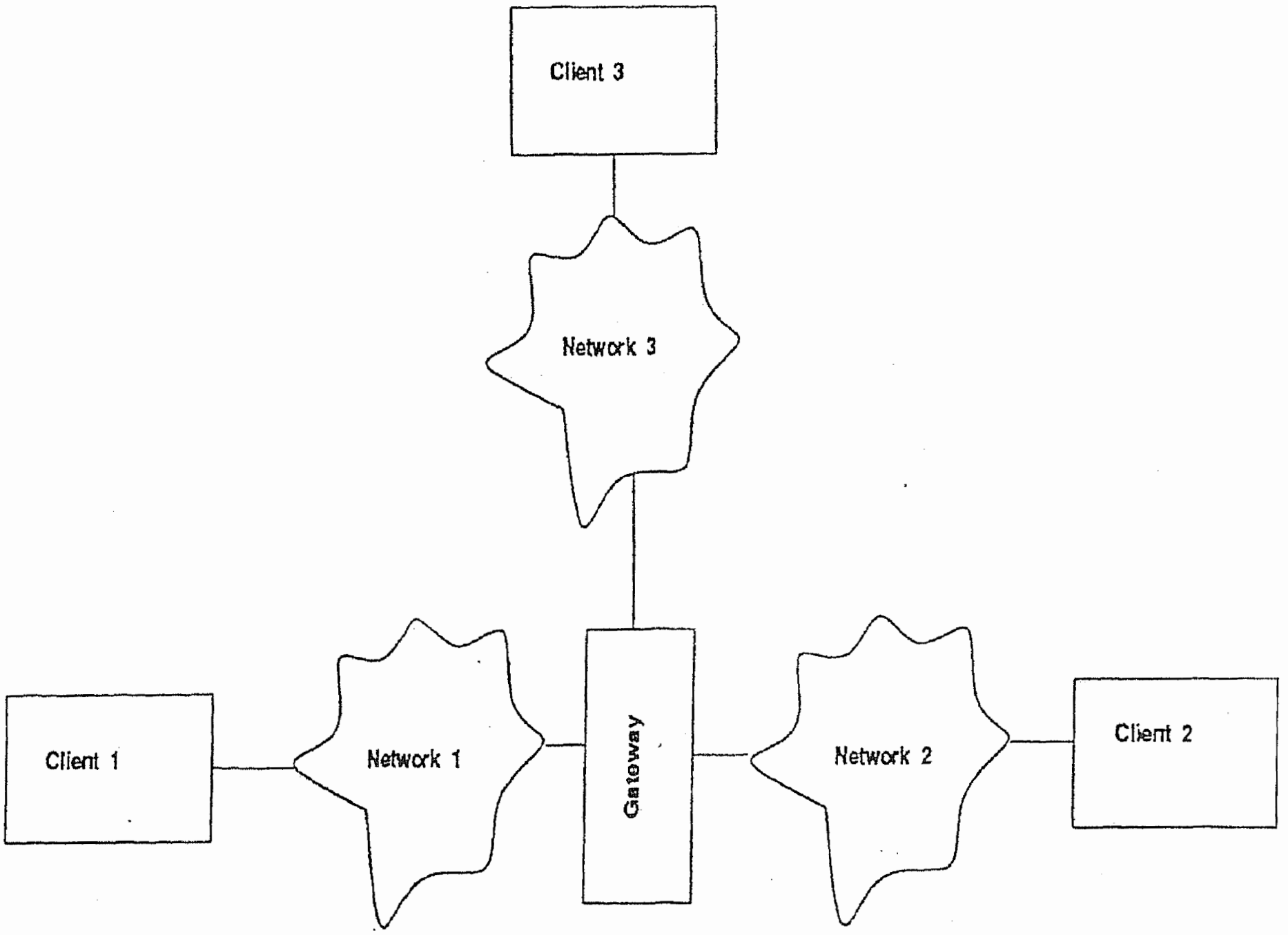

FIG. 3. An example of an internetwork-based distributed system.

A user at *Client1* asks to communicate with a given QoS, with another user at *Client3*, e.g. in audio conferencing; the QoS manager makes use of the negotiation procedure and finds a system configuration that might support the requested service. The configuration consists of *Client1*, *Network1*, *Gateway*, *Network3*, and *Client3* each of these units commits to a certain level of QoS. During the session, say, *Network1* fails to meet the agreed delay, while *Network 3* has certain amount of resources unused.

Using NRRS, a possible scenario to recover from the violation caused by *Network1* is that *Network1* asks *Network3* to reserve more resources to compensate for the QOS violation. At a later time, if *Network1* has enough resources to meet the initially agreed delay, then it will notify *Network3* which might free the resources used for the previous compensation.

3.2.3. *Protocols to implement the NRRS*

In order to support the NRRS, each participating unit must realize the following three functions.

1. *QoS Violation detection*: The QoS-agent of a unit monitors the levels of QoS it is providing. Upon the detection of a QoS-violation over a certain period, the QoS-agent computes (estimates) a QoS level which would be sustainable in the future; traffic-forecasting techniques may be used to compute this QoS level.[21] Then a QUI is generated.

2. *QoS Renegotiation*: The QoS-agent processes the messages it receives, e.g. *QUI or reduction message, etc.* This processing consists of committing to a higher level of QoS by some additional resources, if possible or deallocating additional resources.

3. *Recovery detection (optional)*: The QoS-agent of the unit that initially issued a *QUI message* may monitor the current load of the unit to check its capability to support the initially agreed level of QoS. Upon detection of such a capability, a *reduction message* is generated, which contains the amount of the last violation that had occurred.

There are three types of protocols for implementing NRRS. They are: a *centralized protocol,* a *ring protocol* using a optimal contribution policy, and a *ring protocol* using an immediate contribution policy.

Centralized NRRS protocol

This protocol is based on interactions between the QoS-manager and its agents; when a QoS-agent detects a QoS-violation, it sends *QUI message* to the QoS-manager. The QoS-manager then checks the available resources of the units involved in the configuration; this is performed by sending *request resource messages* to the QoS-agents. Based on the results of this operation, the QoS-manager then decides on a solution and informs the QoS-agents (by sending *confirmation messages*) of the fact that they have to assign more or less resources to this particular application. If after some time the QoS-agent determines that it can again support from the previously agreed QoS, it sends a *reduction message* to the QoS-manager. The QoS-manager may decide to either continue with the currently operating configuration or resort to the one previously agreed.

NRRS ring protocol using optimal contribution policy

The operation of NRRS ring protocol using an optimal contribution policy is similar, to some extent, to the operation of centralized RRS protocol. They differ mainly in two aspects: (1) The NRRS ring protocol is based on direct interaction between the QoS-agents of the units of the linear configuration, in opposition to the centralized protocol, which is based on interaction between the QoS-manager and QoS-agents; and (2) the functions performed by QoS-manager in the case of the centralized protocol are performed by the QoS-agent of the overloaded unit in the case of the NRRS ring protocol. When a QoS-agent detects a QoS violation, it sends a QUI message along with the violation degree to the neighboring unit; the latter computes the maximum-level QoS, Maximum QoS it is able to provide for service in question, and sends a QUI message along with this information to its neighbouring unit. When the QoS-agent of the overloaded unit finally receives the QoS-violation indication, which has passed around the ring and includes the maximum level of QoS each unit is able to support, it checks whether the 'sum' of these levels of QoS is equal or 'better' than the initially agreed (end-to-end) QoS. If the response is positive, the QoS-agent decides on the level of QoS each unit should support,

based on some optimization factors, puts this information in a confirmation signal, and sends the signal over the ring to the other QoS-agents; upon receipt of the signal, the QoS-agent reserves the resources to satisfy the levels of QoS specified in the signal. If the 'sum' is worse than the initially agreed QoS, the QoS-agent (of the overloaded unit) notifies the user/application who may intimate a renegotiation to degrade the agreed QoS, abort the current service session, or ignore the violation. If some time after adaptation, the overloaded QoS-agent determines that it can again support the previously agreed QoS, it sends *relaxation signal* to the other units on the ring. Upon receipt of this signal, the QoS-agents (who participate in solving the previous violation) may deallocate the resources which were used to solve the previous violation.

NRRS ring protocol using immediate contribution policy

The NRRS ring protocol using an immediate contribution policy is also based on the interaction between the QoS-agents of the units of the linear configuration in question; When a QoS detects a QoS violation it sends a violation signal to the neighboring unit; its QoS-agent reserves resources, if available, to completely compensate the violation. The result of this operation may be (1) a success if the violation problem is solved, (2) a failure if the unit is at its maximum utilization, or (3) a failure with an offer if the unit can reserve resources to compensate only a fraction of violation. In any case, the unit should send a *violation signal* that contains the violation degree (e.g. equal to zero in case (1) above) that remains to be absorbed to its neighboring unit. When the QoS-agent of the overloaded unit receives the violation signal it initially sent, it checks the violation degree the signal contains. If it is different from zero, the QoS-agent notifies the user/application; otherwise, it sends a *confirmation signal* towards the unit to make the reservation of the extra-allocated resources effective.

3.3. *Delay recovery scheme*

The purpose of the DRS is to compensate for a delay violation for each transmitted data unit. If the measured delay value of a given data unit is above the commitment, a *QUI message* is sent together with the data unit in question, and the next unit in the chain may apply a higher QoS.

DRS can be used to react only to delay violations; it is suitable for real-time and multimedia applications with stringent temporal requirements. An example is the telerobotics application described in,[23] where the sensory data has strict constraints on end-to-end delay, but relatively low throughput demands.

4. Mixer algorithm

Consider a case of video conferencing where a group of participants in one area are connected through a low-speed link to the majority of the conference participants who enjoy a high-speed network access. Instead of forcing everyone to use a lower bandwidth with a reduced quality audio encoding, a *mixer* may be placed near the low-bandwidth area. This mixer resynchronizes incoming multimedia packets to reconstruct the constant time spacing generated by the sender, mixes the reconstructed streams into a single stream, translates the encoding to a lower-bandwidth one and forwards the lower bandwidth packet stream across the low-speed link.

These packets might be unicast to a single recipient or multicast on a different address to multiple recipients.

Mixers may be designed for a variety of purposes. An example is a video mixer that scales the images of individual people in separate video streams and composites them into one video stream to simulate a group scene.

Definition

A mixer is defined as an intermediate system that receives packets from one or more sources, possibly changes the data format, combines the packets in some manner and then forwards a new packet. Since the timing among multiple input sources will not generally be synchronized, the mixer will make timing adjustments among the streams and generates its own timing for the combined stream. Thus, all data packets originating from a mixer will be identified as having the mixer as their synchronization source.

4.1. *Application of mixers*

Mixers are placed at low-bandwidth link. They could be considered as 'intermediate systems'. Although this support adds some complexity to the protocol, the need for these functions has been clearly established by experiments with multicast audio and video applications in the Internet.

General description

A mixer connects two or more transport-level 'clouds'. Typically, each cloud is defined by a common network and transport protocol (e.g., IP/UDP), multicast address or pair of unicast addresses, and transport-level destination port. (Network-level protocol such as IP version 4 to IP version 6, may be present within cloud invisibly to RTP). One system may serve as a mixer for a number of sessions, but each is considered a logically separate entity.

In order to avoid creating loops when a mixer is installed, the following rules must be observed:

- Each of the clouds connected by mixers participating in one session either must be distinct from all others in at least one of these parameters (protocol, address, port), or must be isolated at the network level from others.
- A derivative of the first rule is that there must not be multiple mixers connected in parallel unless by some arrangement they partition the set of sources to be forwarded.
- All end systems communicate through one or more mixers to share the same SSRC (synchronization source identifier) space, that is SSRC identifiers must be unique among all the end systems. An SSRC is required since mixer resynchronizes the incoming packets from different sources when it is having its own SSRC identifier. SSRC identifiers are kept unique and loops are detected.

There are many varieties of mixers designed for different purposes and applications. Some examples are to add or remove encryption, change the encoding of the data or underlying protocols, or replicate between a multicast address and one or more unicast addresses.

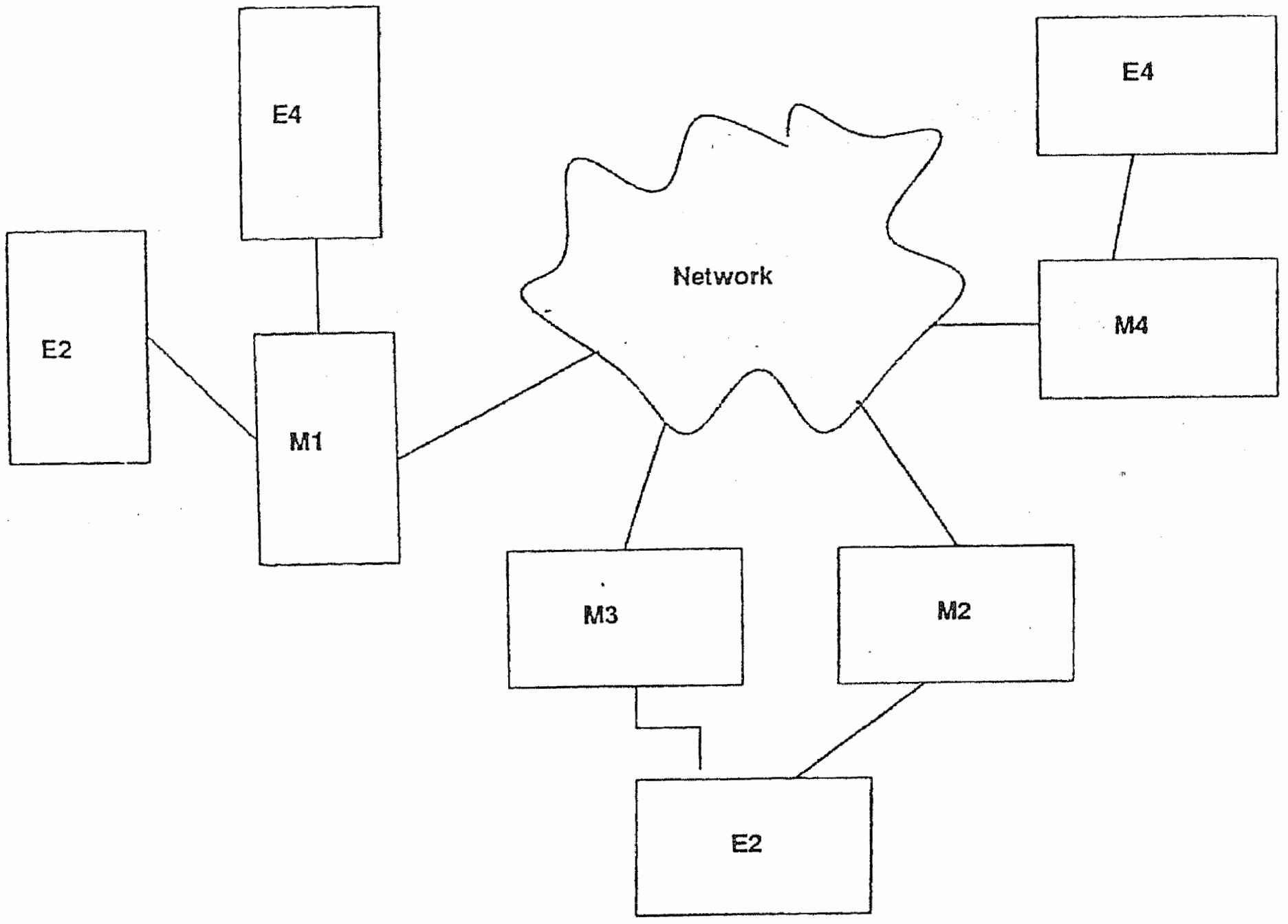

FIG. 4. Sample network with end systems and mixers. M_i = Mixer, i, E_j = End system.

Since the timing among multiple input (in a group scenario) sources may not be synchronized, the mixer will make timing adjustments among the streams and generate its own timing for the combined stream, so it is the synchronization source. Thus all data packets forwarded by a mixer will be marked with the mixer's own SSRC identifier. In order to preserve the identity of the original sources contributing to the mixed packet, CSRC (contributing sources identifiers) list follows the fixed header of the packet. A mixer which is also a contributing source for some packet should explicitly include its own SSRC identifier in the CSRC list for that packet.

A simple network with end systems and mixers is shown in Fig. 4.

Cascaded mixers

If two mixers are cascaded, such as M2 and M3 in Fig. 4, packets received by a mixer may already have been mixed and may include an CSRC list with multiple identifiers. The second mixer should build the CSRC list for outgoing packet using the CSRC identifiers from already mixed input packets and the SSRC identifiers from unmixed input packets.

Algorithm

The function of the proposed mixer algorithm is as follows:

Step 1: Upon receiving an up call from user the Mixer at Base-station intercepts the call and establishes a two-way TCP connection with the specified host of the user.

Step 2: It receives packets from the server and changes the data format of the received packet and forwards it to the user connected to the server with a low bandwidth link (wireless link).

5. Implementation

5.1. *QoS adaptation algorithm*

The proposed *Quality of Service Adaptation* and *Mixer* algorithms are implemented on a wired and wireless network as shown in Fig. 5.

The simulation is done on a wired/wireless network available in PET-Unit (Protocol Engineering and Technology Unit). The network has the following configuration AGNI which acts as a base station for AKASH and VAYU, whereas PCLAB acts as a base station for JALA. All systems use Linux, except SAM, which runs on using Unix. We used Berkeley sockets and Linux system calls to communicate among the hosts.

In our implementation of QoS adaptation algorithm, we made hosts SAM and PROTOCOL as remote servers and host JALA as client (user). Two servers are opened on PROTOCOL and SAM. One server is for video and other for audio.

The two servers on SAM can be adapted by PROTOCOL, if the server on PROTOCOL is busy and vice versa. We used a distributed QoS Manager, in which the QoS Manager acts at both the server and client.

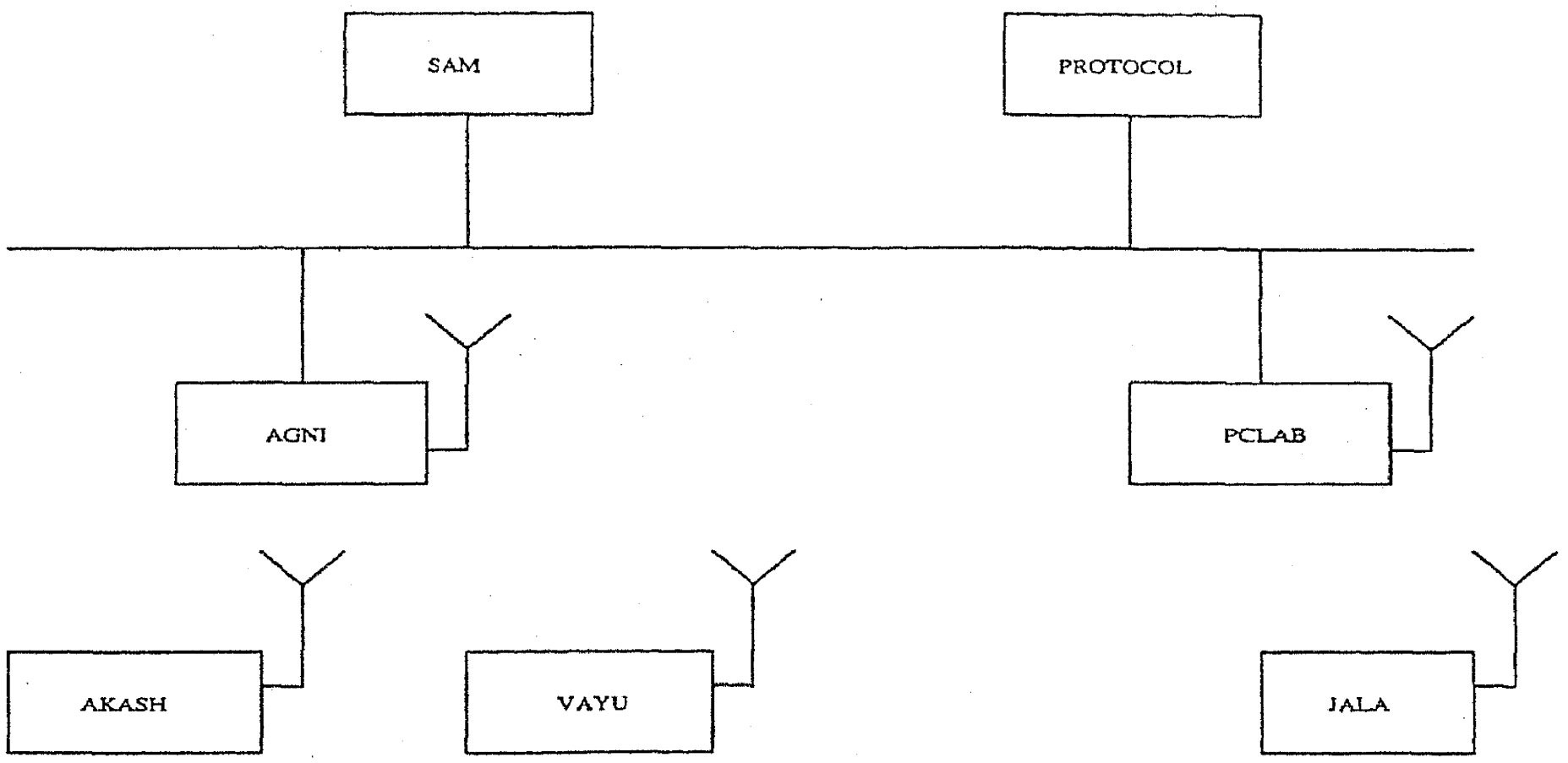

FIG. 5. Network environment for simulation.

The simulation has been carried with various users requesting for multiple data streams. For example, a user at JALA establishes a connection with PROTOCOL, then the system prompts the user to choose the type of application:

1. Video only
2. Audio only
3. Both audio and video.

Depending upon the user's interest and availability of resources (like buffers, speakers, etc.) the user chooses the type of application. Synchronization is done in case if the user chooses both audio and video to play. After choosing the application, the user will be prompted for the QoS parameters like frame-rate, delay, color/gray, size of display. These QoS parameters are intimated by the QoS manager at the client to the QoS manager at the server. The QoS manager at the server gathers information from its resources and enquires with the QoS manager of the client if it is accepting the user's request or not. In our case the request given by JALA is checked by the QoS manager at PROTOCOL, if it is unable to satisfy the user request it checks in its database whether there is any server having the desired application and the QoS it can provide at that instant of time. If the server at the PROTOCOL could find a server on a different system for the purpose then it will adapt to that server to serve the application. Otherwise, the server will prompt the user with degradable service it can offer; if the user accepts the degradable service it will serve the application else it will free the resources allocated for the user and drops the call.

5.2. *Mixer algorithm*

Mixers are also implemented on the same LAN as shown in Fig. 5, and in the same environment as of QoS adaptation implementation. In this case, a Mixer is placed at the PCLAB basestation. The Mixer establishes a two-way TCP connection between the remote server (at SAM) and the client (at JALA) and changes the incoming format of the requested application from remote server and transmits to the user (at JALA).

6. Simulation

We have simulated the QoS adaptation scheme and mixer algorithm on a wired/wireless network in which the client is mobile and the remote server is on wired network, the LAN configuration of which we tested is given in Fig. 5. We have initiated several clients on the host JALA and on other hosts in the network, and the hosts SAM and PROTOCOL act as remote servers serving the application required by the clients.

We have triggered some QoS violations by increasing load and configuration changes in the network and shown the difference in time when the server is busy (enough Resources are not available) and when free (enough resources are available). The time difference of getting the application simulates the case of delay in receiving a packet and hence getting a degraded service (provided we take the maximum acceptable delay by a client as that of delay when the server is free, i.e. enough resources are available). We have also shown, in various cases, the improvement in delay when the QoS manager adapts to the new server in view of non-

Table I
Three users without adaptation

Application	Size (kbytes)	Time take in seconds to serve		
		user 1	user 2	user 3
Tired_Ozzy.au	1802	40	49	48
Right_now.au	2468	70	67	71
Cemetery_gates.au	3506	88	94	93

availability of enough resources. Here some results are tabulated for a case which involves three simultaneous users.

Case 1: *Three users requesting the same resource*

Let there be three users, logged in different hosts, requesting for three audio files for their playout. User 1 requests the server to deliver the file Tired-ozzy.au with maximum delay of 35 s, user 2 sets the maximum tolerable delay at 65 s for the server to get the file right-now.au, and 80 s for the third application to get the file cemetery-gates.au.

Table I shows the average delays experienced by the applications under various network load conditions. Here we can clearly observe that the server is unable to satisfy any of the users request well within the agreed QoS (i.e. delay).

Case 2: *Three users requesting the same resource with adaptation*

In this case the scheduling has been done the same way as in Case 1, requesting the server to supply three audio files to the three users. Due to adaptation algorithms, the two user requests are scheduled to one server and the third user is adapted to other server by QoS manager to deliver the files before the acceptable delay.

Table II shows the average delays in delivering the required files for the users under different network conditions. Here we see two users getting the required multimedia data with the desired QoS unlike the case of without adaptation.

7. Conclusions

In this work, we propose QoS adaptation algorithms that are supporting multimedia in a case of wired and wireless networks and implemented them using Berkeley socket system calls. We also provide a user-friendly menu which prompts the user for the type of application and the

Table II
Three users by adapting the third user

Application	Size (kbytes)	Time take in seconds to serve		
		user1	user2	user3
Tired_Ozzy.au	1802	35	47	32
right_now.au	2468	63	64	38
cemetery_gates.au	3506	86	81	51

desired multimedia data from a remote server with the desired QoS such as frame-rate, colour, etc. It allows to recover *automatically* from QoS violations, and *only require user/application intervention* when the system does not have enough resources to support the current load. We also implemented mixers which can be used at basestations to change the format of incoming stream and transmit it to mobile users in case of inability to support the desired QoS because of bad atmospheric conditions. Both the mixers and QoS adaptation scheme are implemented on TCP/IP suite and found a significant improvement in performance of system to QoS violations and timely delivery of data.

References

1. VENKATARAM, P. — *Current development in multimedia wireless networks*, ICCC, 1995.
2. DESIMONE, A., CHUATH, M. G. AND YUE, O. — Throughput performance of transport-layer protocols over wireless LANS, *Proc. IEEE GlobeComm*, 1993.
3. BANTZ, D. AND BAUCHOT, F. — Wireless LAN design alternatives, *IEEE Network Mag.*, 1994, **8**, 43–53.
4. COX, D. C. — Universal portable radio communications, *IEEE Commun. Mag.*, 1992, pp. 96–115.
5. DUCHAMO, D. AND REYNOLDS, N. F. — Measured performance of a wireless Lan, *Proc. 17th Conf. on Local Computer Networks*, September 1992, pp. 494–499.
6. KAHN, J. J. M. — Wireless LAN design alternatives, *IEEE Network Mag.*, March/April 1994, **8**, 43–53.
7. VENKATARAM, P. — Distributed multimedia applications of emerging personal communication systems, *Proc. IETE*, Hyderabad, Oct. 1–2, 1993.
8. GRILLO, D., CHIA, S. AND RUILEA, N. — The European path towards advanced mobile systems, *IEEE Pers. Commun.*, Feb. 1995, **2**.
9. RAYCHAOUDHARI, D. AND WILSON, N. D. — ATM based transport architecture for multiservices wireless personal communication networks, *IEEE J.*, **SAC-12**, 1994, 8.
10. BOCHINANN, G. AND HAFID, A. — Some principles for QoS management distributed system engineering, *Multimedia Systems*, 1997, **4**(1), 16–27.
11. DANTHINE, A. — High-performance protocol with multimedia support on HSLANs and B-ISDN, *Proc. 3rd Joint European Networking Conf.*, Innsbruck, Austria, 1992.
12. KALKBRENNER, G. — Quality of service in distributed hypermedia systems, *Proc. 2nd Int. Workshop on Principles of Document Processing*, 1994.
13. WALLCE, G. K. — The JPEG still picture compression, *Commun. ACM*, 1991, **34**, 30–44.
14. KARMOUCH, A. — A multimedia medical communications system, *IEEE J.*, 1990, **SAC-8**, 325–339.
15. HAFID, A. AND BOCHMANN, G. — A QOS negotiation procedure for distributed multimedia presentational applications, *Proc. 5th IEEE Int. High Speed Distributed Computing* (HPDC-5), Syracuse, New York, 1996.
16. DELGROSSI, L. — Media scaling for audiovisual communication with the Heidelberg transport system, *Proc. ACM Multimedia*, 1993, pp. 99–104.

17. LIOU, M. L. — Overview of the P× 64 Kbit/s video coding standard, *Commun. ACM*, 1991, **34**, 59–63.

18. NAGARAJAN, R. — QoS issues in high-speed networks, Ph.D. Thesis. University of Massachusetts, 1993.

19. NAHRSTEDT, K. — An architecture for end-to-end QoS provision and its experimental validation, Ph.D Thesis, University of Pennsylvania., 1995.

20. SKELLY, P. AND SCHWARTZ, M. — A cell and burst level control framework for integrated video and image traffic, *IEEE INFOCOM*, June 1994.

21. VENKATARAM, P. — A session protocol for wireless communications, Nortel Report, February 1998.

22. RAMANATHAN, R. AND STEENSTRUP, M. — Hierarchically-organized, multihop mobile wireless networks for quality-of-service support, Mobile Networks and Applications, **3**.

23. CACERES, R. AND IFTODE, L. — The effects of mobility on reliable transport protocols, Technical Report MITL-TR-73-93, MIT Laboratory, Princeton, New Jersey, November 1993.

24. KELLER, R. AND EFFESLBERG, W. — MCAM: An application layer protocol for movie control access, and management, *Proc. 1st ACM Int. Conf. on Multimedia*, ACM Press, 1993.

25. SCHWARTZ, M. — *Telecommunications networks: 1987 protocols, modeling, analysis*, 1987.

26. SCHWARTZ, M. — Network management and control issues in multimedia wireless networks, *IEEE Personal Commun.*, June 1995.

27. NANDA, S., EJZAK, R. AND DOSHI, B. T. — A retransmission scheme for circuit-mode data on wireless links, *IEEE J.*, **SAC-12**, 1994.

29. SOMALINGAM, R. — Network performance monitoring for multimedia networks, Master Thesis, School of Computer Science, McGill University, Montreal, 1996.

30. CHU, T. S. AND GANS, M. J. — High speed infrared local wireless communication, *IEEE Commun.*, Aug. 1987, **25**, 4–10.

31. JACOBSON, V. — Congestion avoidance and control process, ACM SIGCOMM, August 1998, pp. 314–329.

32. VOGEL, A., KERHERVE, B., BOCHMANN, G. V. AND GECSEI, J. — Distributed multimedia and QoS: A survey, *IEEE Multimedia*, Summer 1995.