Multimedia Systems (1998) 6: 138–151

Multimedia Systems

A survey of QoS architectures

Cristina Aurrecoechea, Andrew T. Campbell, Linda Hauw

Center for Telecommunication Research, Columbia University, New York, NY 10027, USA; http://www.ctr.columbia.edu/comet/members.html;
e-mail: {cris,campbell,linda}@ctr.columbia.edu

Abstract. Over the past several years there has been a considerable amount of research within the field of quality-of-service (QoS) support for distributed multimedia systems. To date, most of the work has been within the context of individual architectural layers such as the distributed system platform, operating system, transport subsystem and network layers. Much less progress has been made in addressing the issue of overall end-to-end support for multimedia communications. In recognition of this, a number of research teams have proposed the development of QoS architectures which incorporate QoS-configurable interfaces and QoS driven control and management mechanisms across all architectural layers. This paper examines the state-of-the-art in the development of QoS architectures. The approach taken is to present QoS terminology and a generalized QoS framework for understanding and discussing QoS in the context of distributed multimedia systems. Following this, we evaluate a number of QoS architectures that have emerged in the literature.

1 Introduction

Meeting Quality-of-Service (QoS) guarantees in distributed multimedia systems is fundamentally an end-to-end issue, that is, from application to application. Consider, for example, the remote playout of a sequence of audio and video: in the distributed system platform, QoS assurances should apply to the complete flow of media from the remote server across the network to the point/s of delivery. As illustrated in Fig. 1, this generally requires end-to-end admission testing and resource reservation in the first instance, followed by careful co-ordination of disk and thread scheduling in the end-system, packet/cell scheduling and flow control in the network and, finally, active monitoring and maintenance of the delivered QoS. A key observation is that for applications relying on the transfer of multimedia and, in particular, continuous media flows, it is essential that QoS is configurable, predictable and maintainable system-wide, including the end-system devices, communications subsystem and networks. Furthermore, it is also important that all end-to-end elements of distributed-systems architecture work in unison to achieve the desired application level behavior.

To date, most of the developments in the area of QoS support have occurred in the context of individual architectural components [20]. Much less progress has been made in addressing the issue of an overall QoS architecture for multimedia communications. There has been, however, considerable progress in the separate areas of distributed-systems platforms [20–28], operating systems [29–35], transport systems [36–45] and multimedia networking [46–66] support for QoS. In end-systems, most of the progress has been made in the areas of scheduling [11, 12, 31], flow synchronisation [18, 19] and transport support [36–45]. In networks, research has focused on providing suitable traffic models [2] and service disciplines [52], as well as appropriate admission control and resource reservation protocols [48, 51, 53]. Many current network architectures, however, address QoS from a provider's point of view and analyze network performance, failing to comprehensively address the quality needs of applications. Until recently, there has been little work on QoS support in distributed systems platforms. What work there is has been mainly carried out in the context of the open distributed processing [27].

The current state of QoS support in architectural frameworks can be summarized as follows [20]:

i) *incompleteness*: current interfaces (e.g., application programming interfaces such as Berkeley Sockets) are generally not QoS configurable and provide only a small subset of the facilities needed for control and management of multimedia flows;
ii) *lack of mechanisms to support QoS guarantees*: research is needed in distributed control, monitoring and maintenance QoS mechanisms, so that contracted levels of service can be predictable and assured; and
iii) *lack of an overall framework*: it is necessary to develop an overall architectural framework to build upon and reconcile the existing notion of QoS at different system levels and among different network architectures.

Correspondence to: C. Aurrecoechea

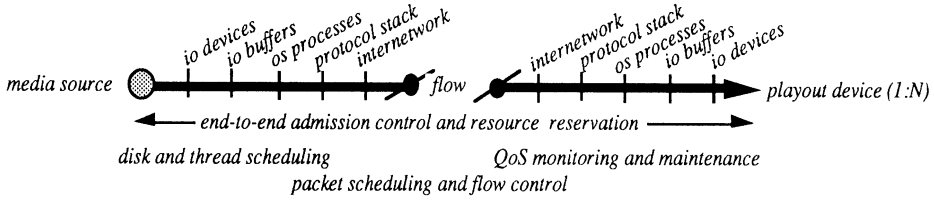

Fig. 1. End-to-end QoS scenario for a continuous media flow

In recognition of the above limitations, a number of research teams have proposed systems architectural approaches to QoS support. In this paper, these are referred to as *QoS architectures* [67–90]. The intention of QoS architecture research is to define a set of QoS configurable interfaces that formalize QoS in the end-system and network, providing a framework for the integration of QoS control and management mechanisms.

In this paper, we present, in Sect. 2, a *generalized QoS framework* and terminology[1] for distributed multimedia applications operating over multimedia networks with QoS guarantees. The generalized QoS framework is based on a set of principles that govern the behavior of QoS architectures. Following this, we evaluate a number of QoS architectures found in the literature that have been developed by the telecommunications, computer communications and standards communities. We then present a short qualitative comparison and discussion in Sects. 4 and 5, respectively. Finally, in Sect. 6 we offer some concluding remarks.

2 Generalized QoS framework

In what follows, a set of elements used in building QoS into distributed multimedia systems is described. This includes QoS principles which govern the construction of a generalized QoS framework, QoS specification which captures application-level QoS requirements, and QoS mechanisms which realize the desired application end-to-end QoS behavior.

2.1 QoS principles

A number of QoS principles motivate the design of a generalized QoS framework:

- *transparency principle* states that applications should be shielded from the complexity of underlying QoS specification and QoS management. An important aspect of transparency is the QoS-based API [74, 9] at which desired QoS levels are stated (see QoS management policy in Sect. 2.2). The benefits of transparency are that it reduces the need to embed functionality in the application, hides the detail of underlying service specification from the application and it delegates the complexity of handling QoS management activities to the underlying framework;
- *integration principle* states that QoS must be configurable, predictable and maintainable over all architectural layers to meet end-to-end QoS [68]. Flows[2] traverse resource modules (e.g., CPU, memory, multimedia devices, network, etc.) at each layer from source media devices, down through the source protocol stack, across the network, up through the receiver protocol stack to the playout devices. Each resource module traversed must provide QoS configurability (based on a QoS specification), resource guarantees (provided by QoS control mechanisms) and maintenance of ongoing flows;
- *separation principle* states that media transfer, control and management are functionally distinct architectural activities [69]. The principle states that these tasks should be separated in architectural QoS frameworks. One aspect of this separation is the distinction between signaling and media transfer. Flows (which are isochronous in nature) generally require a wide variety of high-bandwidth, low-latency, non-assured services with some form of jitter correction. On the other hand, signaling (which is full duplex and asynchronous in nature) generally requires low-bandwidth, assured-type services;
- *multiple time scales principle* [69] guides the division of functionality between architectural modules and pertains to the modeling of control and management mechanisms. It is necessitated by, and is a direct consequence of, fundamental time contraints that operate in parallel between resource management activities (e.g., scheduling, flow control, routing, QoS management, etc.) in distributed communications environments; and
- *performance principle* subsumes a number of widely agreed rules for the implementation of QoS-driven communications systems which guide the division of functionality in structuring communication protocols for high performance in accordance with systems design principles [6], avoidance of multiplexing [7], recommendations for structuring communications protocols [8], and the use of hardware assists for efficient protocol processing [40, 55].

2.2 QoS specification

QoS specification is concerned with capturing application-level QoS requirements and management policies. QoS specification is generally different at each system layer and is

[1] Where appropriate, we have adopted the standard terminology of the ISO QoS Working Group [67].

[2] The notion of a flow is an important abstraction which underpins the development of QoS frameworks. Flows characterize the production, transmission and eventual consumption of a single media source (viz. audio, video, data) as integrated activities governed by single statements of end-to-end QoS. Flows are simplex in nature and can be either unicast or multicast. Flows generally require end-to-end admission control and resource reservation, and support heterogeneous QoS demands.

used to configure and maintain QoS mechanisms resident in the end-system and network. For example, at the distributed system platform level, QoS specification is primarily application-oriented rather than system oriented. Lower level considerations such as tightness of synchronisation of multiple related audio and video flows, the rate and burst size of flows, or the details of thread scheduling in the end-system should all be hidden at this level. QoS specification is therefore declarative in nature; applications specify what is required rather than how this is to be achieved by underlying QoS mechanisms. QoS specification encompasses but is not limited[3] to the following:

- *flow performance specification*, which characterizes the user's flow performance requirements [5]. The ability to guarantee traffic throughput rates, delay, jitter and loss rates is particularly important for multimedia communications. These performance-based metrics are likely to vary from one application to another. To be able to commit necessary end-system and network resources, QoS frameworks must have prior knowledge of the expected traffic characteristics associated with each flow before resource guarantees can be met;
- *level of service*, which specifies the degree of end-to-end resource commitment required (e.g., deterministic [49], predictive [47] and best effort [8]). While the flow performance specification permits the user to express the required performance metrics in a quantitative manner, level of service allows these requirements to be refined in a qualitative way to allow a distinction to be made between hard and soft performance guarantees. Level of service expresses a degree of certainty that the QoS levels requested at the time of flow establishment or renegotiation will be honored;
- *QoS management policy*, which captures the degree of QoS adaptation [74] that the flow can tolerate and the scaling actions to be taken in the event of violations in the contracted QoS [86]. By trading off temporal and spatial quality to available bandwidth, or manipulating the playout time of continuous media in response to variation in delay, audio and video flows can be presented at the playout device with minimal perceptual distortion. The QoS management policy also includes application-level selection for QoS indications (aka QoS alerts [67]) in the case of violations in the requested QoS and periodic QoS availability notifications for bandwidth, delay, jitter and loss;
- *cost of service*, which specifies the price the user is willing to incur for the level of service [10]. Cost of service is a very important factor when considering QoS specification. If there is no notion of cost of service involved in QoS specification, there is no reason for the user to select anything other than maximum level of service, e.g., guaranteed service; and
- *flow synchronization specification*, which characterizes the degree of synchronisation (i.e., tightness) between multiple related flows [18]. For example, simultaneously recorded video perspectives must be played in precise frame-by-frame synchrony so that relevant features may be simultaneously observed. On the other hand, lip synchronization in multimedia flows does not need to be absolutely precise [19] when the main information channel is auditory and video is only used to enhance the sense of presence.

[3] Note that QoS specification could also include other important areas such as security. However, we do not deal with security in this paper.

2.3 QoS mechanisms

QoS mechanisms are selected and configured according to user-supplied QoS specification, resource availability and resource management policy. In resource management, QoS mechanisms can be categorized as either static or dynamic in nature. *Static resource management* deals with flow establishment and end-to-end QoS re-negotiation phases (which we describe as QoS provision) and *dynamic resource management* deals with the media-transfer phase (which we describe as QoS control and management). The distinction between QoS control and QoS management is characterized by the different time scales over which they operate. QoS control operates on a faster time scale than QoS management.

2.3.1 QoS provision mechanisms

QoS provision is comprised of the following components:

- *QoS mapping*, which performs the function of automatic translation between representations of QoS at different system levels (i.e., operating system, transport layer, network, etc.) and thus relieves the user of the necessity of thinking in terms of lower level specification. For example, the transport-level QoS specification may express flow requirements in terms of level of service, average and peak bandwidth, jitter, loss and delay constraints. For admission testing and resource allocation purposes, this representation must be translated to something more meaningful to the end-system. As illustrated in Fig. 2, QoS mapping derives the scheduler QoS parameters (viz. period, quantum and deadline times of the threads) from the transport-level QoS specification parameters [34];
- *admission testing*, which is responsible for comparing the resource requirement arising from the requested QoS against the available resources in the system. The decision whether a new request can be accommodated generally depends on system-wide resource management policies and resource availability. Once admission testing has been successfully completed on a particular resource module, local resources are reserved immediately and then committed later if the end-to-end admission control test (i.e., accumulation of hop-by-hop tests) is successful; and
- *resource reservation protocols*, which arrange for the allocation of suitable end-system and network resources according to the user QoS specification. In doing so, the resource reservation protocol interacts with QoS-based routing to establish a path through the network in the first instance, then, based on QoS mapping and admission control at each local resource module traversed (e.g. CPU, memory, I/O devices, switches, routers, etc.), end-to-end resources are allocated. The result is that QoS

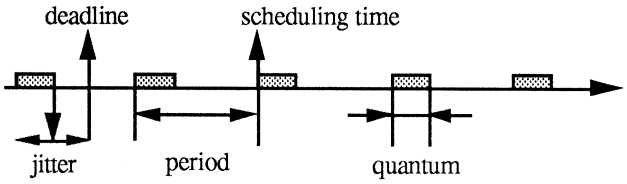

Fig. 2. QoS parameters derived during QoS mapping

control mechanisms such as network-level cell/packet schedulers and end-system thread schedulers are configured accordingly.

2.3.2 QoS control mechanisms

QoS control mechanisms operate on time scales at or close to media transfer speeds. They provide real-time traffic control of flows based on requested levels of QoS established during the QoS provision phase. The fundamental QoS control mechanisms include the following:

- *flow scheduling*, which manages the forwarding of flows (chunks of media based on application-layer framing) in the end-system [30–35] and network (packets and/or cells) in an integrated manner [52]. Flows are generally scheduled independently in the end-systems, but may be aggregated and scheduled in unison in the network. This is dependent on the level of service and the scheduling discipline [2] adopted;
- *flow shaping*, which regulates flows based on user-supplied flow performance specifications. Flow shaping can be based on a fixed-rate throughput (i.e., peak rate) or some form of statistical representation (i.e., sustainable rate and burstiness) of the required bandwidth [49]. The benefit of shaping traffic is that it allows the QoS frameworks to commit sufficient end-to-end resources and to configure flow schedulers to regulate traffic through the end-systems and network. It has been mathematically proven[4] that the combination of traffic shaping at the edge of the network and scheduling in the network can provide hard performance guarantees;
- *flow policing*, which can be viewed as the dual of monitoring. Monitoring, which is usually associated with QoS management, observes whether the QoS contracted by a provider is being maintained, whereas policing observes whether the QoS contracted by a user is being adhered to. Policing is often only appropriate where administrative and charging boundaries are being crossed, for example, at a user-to-network interface [53]. Flow-shaping schemes at the source allow the policing mechanism to detect misbehaving flows;
- *flow control*, which includes both open-loop and closed-loop schemes. Open-loop flow control is used widely in telephony and allows the sender to inject data into the network at the agreed levels, given that resources have been allocated in advance. Closed-loop flow control requires the sender to adjust its rate based on feedback from the receiver [41] or network [64]. Applications using closed-loop flow control based protocols must be able to adapt to fluctuations in the available resources. On the other hand, applications which cannot adjust to changes in the delivered QoS are more suited to open-loop schemes, where bandwidth, delay and loss can be deterministically guaranteed for the duration of the session; and
- *flow synchronization*, which is required to control the event ordering and precise timings of multimedia interactions. Lip-sync is the most commonly cited form of multimedia synchronization (i.e., synchronization of video and audio flows at a playout device). Other synchronization scenarios reported include: event synchronization with and without user interaction, continuous synchronization other than lip-sync, continuous synchronization for disparate sources and sinks. All scenarios place fundamental QoS requirements on flow synchronization protocols [44].

[4] Parekh [56] has shown that, if a source is shaped by a token bucket with leaky bucket rate control and scheduled by the weighted fair-queuing service discipline [58], it is possible to achieve strong guarantees on delay.

2.3.3 QoS management mechanisms

In order to maintain agreed levels of QoS, it is often insufficient to just commit resources. Rather, QoS management is frequently required to ensure that the contracted QoS is sustained. QoS management of flows is functionally similar to QoS control. However, it operates on a slower time scale; that is, over longer monitoring and control intervals [15]. The fundamental QoS management mechanisms include the following:

- *QoS monitoring*, which allows each level of the system to track the ongoing QoS levels achieved by the lower layer. QoS monitoring often plays an integral part in a QoS maintenance feedback loop which maintains the QoS achieved by resource modules. Monitoring algorithms operate over different time scales. For example, they can run as part of a scheduler (as a QoS control mechanism) to measure individual performance of ongoing flows. In this case, measured statistics can be used to control packet scheduling and admission control [47]. Alternatively, QoS monitoring can operate on an end-to-end basis as part of a transport-level feedback mechanism [44] or as part of the application itself [13];
- *QoS availability*, which allows the application to specify the interval over which one or more QoS parameters (e.g., delay, jitter, bandwidth, loss, synchronization) can be monitored and the application informed of the delivered performance via a QoS signal [74]. Both single and multiple QoS signals can be selected based on the user-supplied QoS management policy (see Sect. 2.2);
- *QoS degradation*, which issues a QoS indication to the user when it determines that the lower layers have failed to maintain the QoS of the flow and nothing further can be done by the QoS maintenance mechanism. In response to such an indication, the user can choose either to adapt to the available level of QoS or scale back [85] to a reduced level of service (i.e., end-to-end renegotiation);
- *QoS maintenance*, which compares the monitored QoS against the expected performance and then exerts tun-

ing operations (i.e., fine- or coarse-grain resource adjustments) on resource modules to sustain the delivered QoS. Fine-grain resource adjustment counters QoS degradation by adjusting local resource modules (e.g., loss via the buffer management, throughput via the flow regulation, and queuing delays and continuous-media playout calculation via the flow scheduling [86]); and

- *QoS scalability*, which comprises QoS filtering (which manipulates flows as they progress through the communications system) and QoS adaptation (which scales flows at the end-systems only) mechanisms. Many continuous-media applications exhibit robustness in adapting to fluctuations in end-to-end QoS. Based on the user-supplied QoS management policy, QoS adaptation in the end-systems can take remedial actions to scale flows appropriately. Resolving heterogeneous QoS issues is a particularly acute problem in the case of multicast flows. Here individual receivers may have differing QoS capabilities to consume audio-visual flows; QoS filtering helps to bridge this heterogeneity gap, while simultaneously meeting individual receivers' QoS requirements [90].

3 QoS architectures

Until recently, research in providing QoS guarantees has primarily focused on network-oriented traffic models and service-scheduling disciplines. These guarantees are not, however, end-to-end in nature. Rather, they preserve QoS guarantees only between network access points that end-systems are attached to [81]. Work on QoS-driven end-system architecture needs to be integrated with network-configurable QoS services and protocols to meet application-to-application QoS requirements. In recognition of this, researchers have recently proposed new communication architectures which are broader in scope and cover both network and end-system domains. In this section, we review a number of QoS architectures which have recently emerged in the literature [67–90]. Each architecture tends to use its own distinctive QoS terminology. We do not attempt to resolve that here. We present, rather, the pertinent and novel features of each architecture and then, in Sect. 4, compare them with the generalized QoS framework introduced in the preceding section.

3.1 Heidelberg QoS model

The HeiProject at IBM's European Networking Center in Heidelberg has developed a comprehensive QoS model, which provides guarantees in the end-systems and network [71]. The communications architecture includes a continuous-media transport systems (HeiTS/TP) [42], which provides QoS mapping and *media scaling* [85] as illustrated in Fig. 3. Underlying the transport is an internetworking layer based on ST-II [46], which supports both guaranteed and statistical levels of service. In addition, the network supports QoS-based routing and QoS filtering. Key to providing end-to-end guarantees is *HieRAT (resource administration technique)* [71]. HeiRAT is comprised of a comprehensive

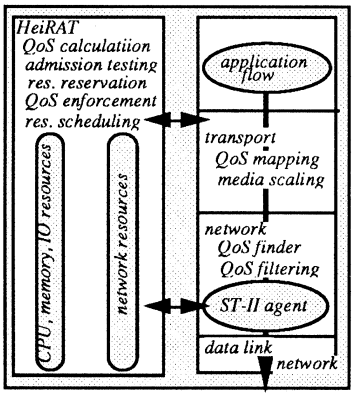

Fig. 3. Heidelberg QoS model

QoS management scheme which includes QoS negotiation, QoS calculation, admission control, QoS enforcement and resource scheduling. The HeiRAT operating system scheduling policy is a rate-monotonic scheme, whereby the priority of a system thread performing protocol processing is proportional to the message rate requested.

The Heidelberg QoS model has been designed to handle heterogeneous QoS demands from individual receivers in a multicast group and to support QoS adaptivity via flow-filtering and media-scaling techniques. Media scaling [85] and codec translation at the end-systems and flow filtering and resource sharing in the network are fundamental to meeting heterogeneous QoS demands. Media scaling matches the source with the receivers' QoS capability by manipulating flows at the network edges. In contrast, filtering accommodates the receivers' QoS capability by manipulating flows at the core of the network as flows traverse bridges, switches and routers.

3.2 XRM

The COMET group at Columbia University is developing an *Extended Integrated Reference Model (XRM)* [28] as a modeling framework for control and management of multimedia telecommunications networks (which comprise multimedia computing platforms and broadband networks). The COMET group argues that the foundations for operability (i.e., control and management) of multimedia computing and networking devices are equivalent; that is, both classes of devices can be modeled as producers, consumers and processors of media.The only difference between computing and network devices is the overall goal that a group of devices has set to achieve in the network or end-system. The XRM is divided into five distinct planes [69] as illustrated in Fig. 4:

- *management function*, which resides in the network management plane (N-plane) and covers the OSI functional areas of network and system management;
- *traffic control function*, which comprises the resource control (M-plane) and connection management and control (C-plane) planes. Resource control constitutes cell scheduling, call admission, call routing in the network, process scheduling, memory management, routing, admission control and flow control in the end-systems;

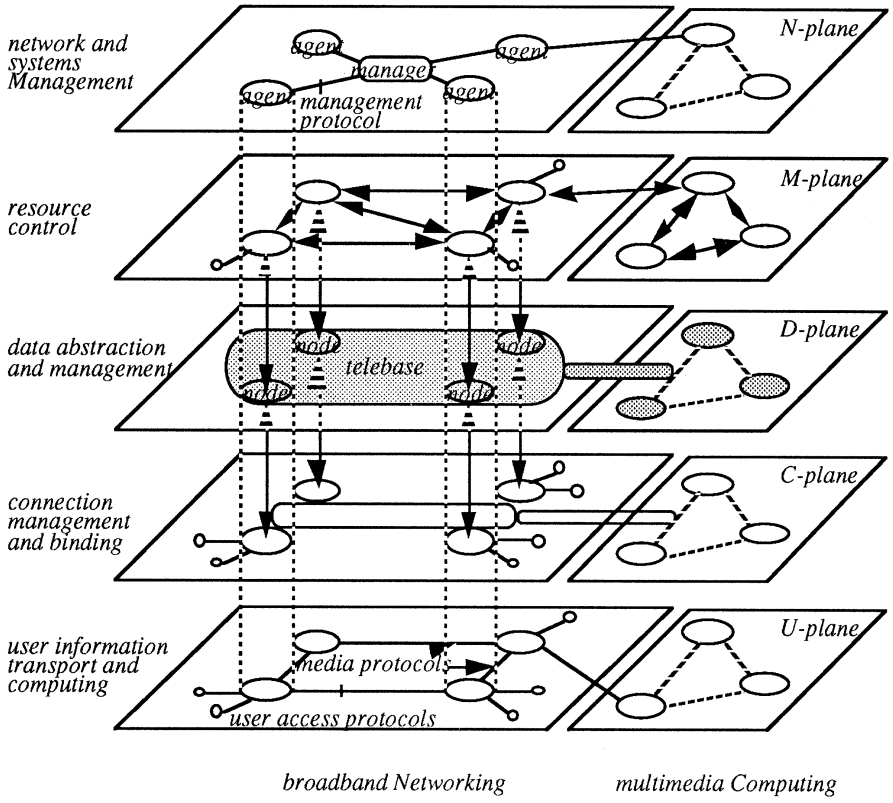

Fig. 4. XRM

- *information transport function*, which is located in the user transport plane (U-plane), models the media protocols and entities for the transport of user information in both the network and the end-systems; and
- *telebase*, which resides in the data abstraction and management plane (D-plane) and collectively represents the information, data abstractions existing in the network and end-systems. The telebase implements data sharing among all other XRM planes.

The XRM is built on theoretical work of guaranteeing QoS requirements in ATM networks and end-systems populated with multimedia devices. General concepts for characterizing the capacity of network [82] and end-system [73] devices (e.g., disks, switches, etc.) have been developed. At the network layer, XRM characterizes the capacity region of an ATM multiplexer with QoS guarantees as a *schedulable region*. Network resources such as switching bandwidth and link capacity are allocated based on four cell-level traffic classes (class I, II, III, and C) for circuit emulation, voice and video, data, and network management, respectively. A traffic class is characterized by its statistical properties and QoS requirements. Typically, QoS requirements reflect cell loss and delay constraints. In order to efficiently satisfy the QoS requirements of the cell level, scheduling and buffer management algorithms dynamically allocate communication bandwidth and buffer space appropriately.

In the end-system, flow requirements are modeled through service class specifications with QoS constraints. For example, in the audio video unit, the service class specification is in terms of JPEG, MPEG-I, MPEG-II video and CD audio quality flows with QoS guarantees. QoS for these classes is specified by a set of frame delay and loss constraints.The methodology of characterizing network resources is extended to the end-system to represent the capacity of multimedia devices. Using the concept of a *multimedia capacity region*, the problem of scheduling flows in the end-system becomes identical to the real-time bin-packing exercise of the network layer. The implementation of XRM including key resource abstractions (viz. schedulable and multimedia capacity region) is currently being realized as part of a *binding architecture* [28] for open signaling, control and management of multimedia networks.

3.3 OMEGA

During the past 3 years the University of Pennsylvania has been developing an end-point architecture called the OMEGA architecture [70]. OMEGA is the result of an interdisciplinary research effort that is examining the relationship between application QoS requirements (which make stringent resource demands) and the ability of local (the operating system) and global resource management (combining communication and remotely managed resources) to satisfy these demands. The OMEGA architecture illustrated in Fig. 5 assumes a network subsystem which provides bounds on delay, errors and can meet bandwidth demands, and an operating system which is capable of providing run-time QoS guarantees. The essence of the OMEGA architecture is resource reservation and management of end-to-end resources. Communication is preceded by a call setup phase, where application requirements, expressed in terms of QoS parameters, are

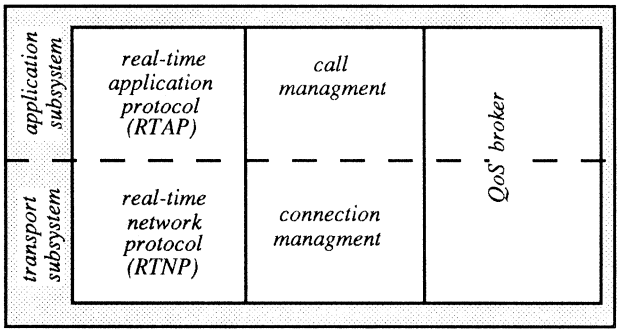

Fig. 5. OMEGA

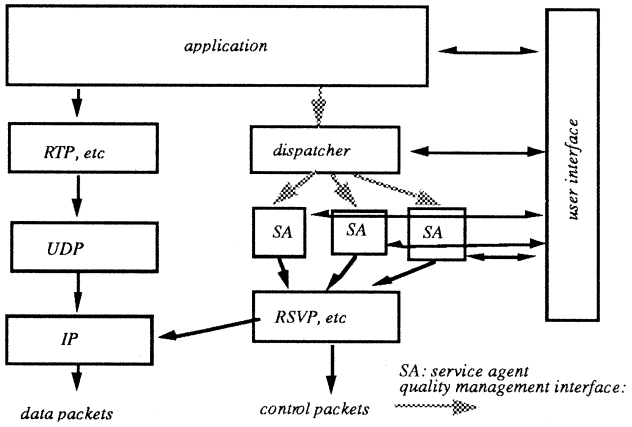

Fig. 6. int-serv QM

negotiated, and guarantees are made at several logical levels, such as between applications and the network subsystem, applications and the operating system, the network subsystem and the operating system. This establishes customized connections and results in the allocation of resources appropriate to meet application requirements and operating system/network capabilities. To facilitate this resource management process, the University of Pennsylvania has also developed a *QoS brokerage model* [88], which incorporates QoS translation, and QoS negotiation and re-negotiation (see [89] for full details on similar work on QoS negotiation protocol at University of Montreal).

3.4 int-serv architecture

The work by the Integrated Services (int-serv) Group [62] of the Internet Engineering Task Force (IETF) is a significant contribution to providing controlled QoS for multimedia applications over an integrated services internetwork. The group has defined a comprehensive int-serv architecture [62] and a QoS framework [79] used to specify the functionality of internetwork system elements (known as elements) which make multiple, dynamically selectable QoS available to applications. The behavior of elements, which constitute routers, subnetworks and end-point operating systems, is captured as a set of services, of which some or all are offered by each element. Each element is QoS-aware and supports interfaces required by the service definition [62]. The concatenation of service elements along an end-to-end data path provides an overall statement of end-to-end QoS. The following int-serv services are offered in addition to best effort: *(i) controlled delay*, which attempts to provide several levels of delay which the application can choose from; *(ii) predicated delay*, which provides a statistical delay bound similar to Tenet Group's statistical service [49] and the COMET Group's guaranteed service [61]; and *(iii) guaranteed delay*, which provides an absolute guaranteed delay bound.

Flows in an int-serv architecture are characterized by two specifications: a *traffic specification*, which is a specification of the traffic pattern which a flow expects to exhibit; and a *service request specification*, which is a specification of the QoS a flow desires from a service elements. The int-serv architecture, which is restricted to the network but also applicable in the end-system, is comprised of four components [62]:

- a *packet scheduler*, which forwards packets streams using a set of queues and timers;
- a *classifier*, which maps each incoming packet into a set of QoS classes;
- an *admission controller*, which implements the admission control algorithm to determine whether a new flow can be admitted or denied; and
- a *reservation setup protocol* (e.g., RSVP [48]), which is necessary to create and maintain the flow-specific state in the routers along the path of the flow.

In [80], Clark introduces some early work on a *Quality-of-Service Manager (QM)* as part of the end-system int-serv architecture. As illustrated in Fig. 6, the QM, which constitutes a user interface, service agents and dispatcher, presents an abstract management layer designed to isolate applications from underlying details of the specific services provided by a QoS-driven internet [62]. One motivating factor behind the introduction of a QM is that applications can negotiate desired QoS without needing to know the details of a specific network service described above. In this case, the QM provides a degree of transparency, whereby applications express desired levels of QoS in application-oriented language rather than using communication QoS specifics. The QM is responsible for determining what QoS management capabilities are available on the application's communication path and chooses the path best suited to the application.

3.5 QoS-A

The *Quality-of-Service Architecture (QoS-A)* [68] is a layered architecture of services and mechanisms for QoS management and control of continuous media flows in multiservice networks. The architecture incorporates the following key notions: *flows*, which characterize the production, transmission and eventual consumption of single media streams (both unicast and multicast) with associated QoS; *service contracts*, which are binding agreements of QoS levels between users and providers; and *flow management*, which provides for the monitoring and maintenance of the contracted QoS levels. The realization of the flow concept demands active QoS management and tight integration between device management, end-system thread scheduling, communications protocols and networks.

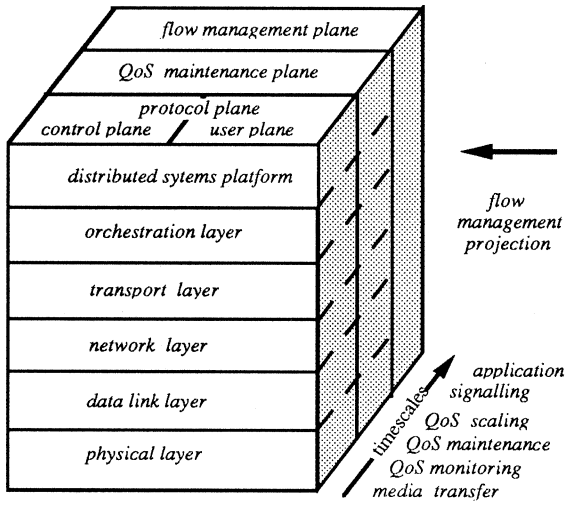

Fig. 7. QoS-A

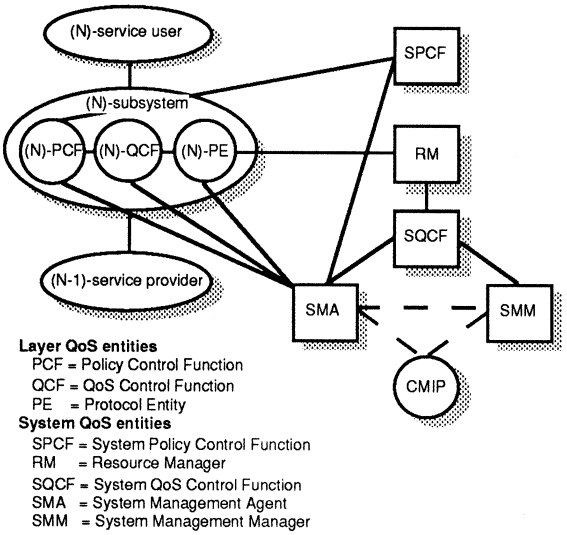

Fig. 8. OSI QoS framework

In functional terms, the QoS-A (as illustrated Fig. 7) is composed of a number of layers and planes. The upper layer consists of a distributed-applications platform augmented with services to provide multimedia communications and QoS specification in an object-based environment [24]. Below the platform level is an orchestration layer which provides jitter correction and multimedia synchronization services across multiple related application flows [44]. Supporting this is a transport layer which contains a range of QoS-configurable services and mechanisms [34]. Below this, an internetworking layer and lower layers form the basis for end-to-end QoS support.

QoS management is realized in three vertical planes in the QoS-A. The protocol plane, which consists of distinct user and control subplanes, is motivated by the principle of separation. QoS-A uses separate protocol profiles for the control and media components of flows because of the different QoS requirements of control and data. The QoS maintenance plane contains a number of layer-specific QMs. These are each responsible for the fine-grained monitoring and maintenance of their associated protocol entities. For example, at the orchestration layer [44], the QM is interested in the tightness of synchronization between multiple related flows. In contrast, the transport QM is concerned with intra-flow QoS such as bandwidth, loss, jitter and delay. Based on flow monitoring information and a user-supplied service contract, QMs maintain the level of QoS in the managed flow by means of fine-grained resource-tuning strategies. The final QoS-A plane pertains to flow management, which is responsible for flow establishment (including end-to-end admission control, QoS-based routing and resource reservation), QoS mapping (which translates QoS representations between layers) and QoS scaling (which constitutes QoS filtering and QoS adaptation for coarse-grained QoS maintenance control).

3.6 OSI QoS framework

One early contribution to the field of QoS-driven architecture is the *OSI QoS Framework* [67], which concentrates primarily on QoS support for OSI communications. The OSI framework broadly defines terminology and concepts for QoS and provides a model which identifies objects of interest to QoS in open system standards. The QoS associated with objects and their interactions is described through the definition of a set of QoS characteristics. The key OSI QoS framework concepts include:

- *QoS requirements*, which are realized through QoS management and maintenance entities;
- *QoS characteristics*, which are a description of the fundamental measures of QoS that have to be managed;
- *QoS categories*, which represent a policy governing a group of QoS requirements specific to a particular environment such as time-critical communications; and
- *QoS management functions*, which can be combined in various ways and applied to various QoS characteristics in order to meet QoS requirements.

The OSI QoS framework (as illustrated in Fig. 8) is made up of two types of management entities (viz. *layer-specific* and *system-wide entities*) that attempt to meet the QoS requirements by monitoring, maintaining and controlling end-to-end QoS. The task of the policy control function is to determine the policy which applies at a specific layer of an open system. The policy control function models any priority actions that must be performed to control the operation of the layer. The definition of a particular policy is layer-specific and therefore cannot be generalized. Policy may, however, include aspects of security, time-critical communications and resource control. The role of the QoS control function is to determine, select and configure the appropriate protocol entities to meet layer-specific QoS goals. The system management agent is used in conjunction with OSI systems management protocols to enable system resources to be remotely managed. The local resource manager represents end-system control of resources. The system QoS control function combines two system-wide capabilities: to tune performance of protocol entities and to modify the capability of remote systems via OSI systems management. The OSI systems management interface is supported by the systems

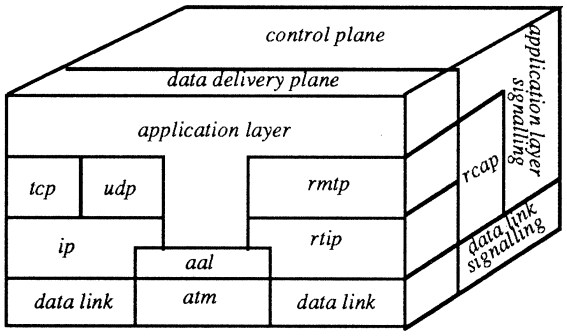

Fig. 9. Tenet Architecture

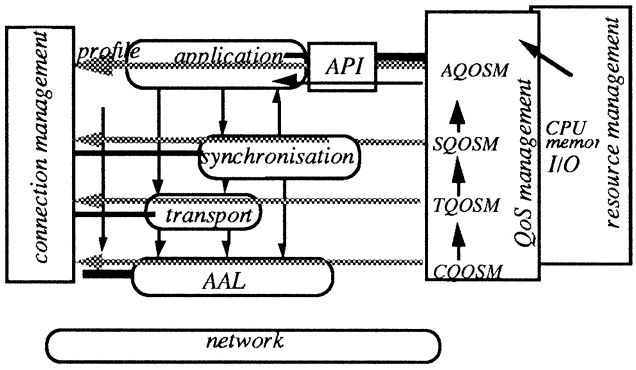

Fig. 10. MASI schematic

management manager, which provides a standard interface to monitor, control and manage end-systems. The system policy control function interacts with each layer-specific policy control function to provide an overall selection of QoS functions and facilities.

3.7 Tenet architecture

The Tenet Group at the University of California at Berkeley has developed a family of protocols [37, 49] which run over an experimental wide-area ATM network. As illustrated in Fig. 9, the *Tenent Architecture* [84] includes a Real-Time Channel Administration Protocol (RCAP) [51] in addition to Real-Time Internet Protocol (RTIP), Continuous Media Transport Protocol (CMTP) [37]. The former provides generic connection establishment, resource reservation and signaling functions for the rest of the protocol family. RCAP spans the transport and network layers for overall resource reservation and flow setup. CMTP is explicitly designed for continuous media support. It is a lightweight protocol which runs on top of RTIP and provides sequenced and periodic delivery of continuous media samples with QoS control over throughput, delays and error bounds. The Tenet Group makes a distinction between deterministic and statistical guarantees for hard real-time and continuous media flows [50], respectively. In the deterministic case, guarantees provide a hard bound on the performance of all cells within a session. Statistical guarantees promise that no more than x% of packets would experience a delay greater than specified, or no more that x% of cells might in a session might be lost.

3.8 TINA QoS framework

The TINA architecture is governed by the separation between telecommunication applications and the TINA Distributed Processing Environment (TDPE). The TDPE software can be visualized as a distributed operating system layer which supports the execution of telecommunication applications. Multimedia services offered by a provider utilize the TDPE and underlying computing and communications capabilities. The TINA QoS Framework [76] addresses the specification and realization of QoS support for telecommunications applications. The framework is partly based on the ANSA [27] and CNET QoS Frameworks [26]. QoS specification is stated declaratively, using the notion of *service attributes* as part of the computational specification. This has been integrated into the TINA-ODL specification language which provides extensions to OMG-IDL [75]. QoS mechanisms have been specified as part of the TDPE specification for QoS provision and QoS negotiation. These mechanisms consider QoS mapping from the application level to the QoS offered by the TDPE kernel and QoS degradation reports in the case that the contracted QoS fails to meet its agreed targets.

3.9 MASI end-to-end model

The CESAME Project [77] at Laboratoire MASI, Université Pierre et Marie Curie, is developing an architecture for multimedia communications which takes end-to-end QoS support as it primary objective. As with the QoS-A, the MASI architecture (shown in Fig. 10) offers a generic QoS framework to specify and implement the required QoS requirements of distributed multimedia applications operating over ATM-based networks. The CESAME Project considers end-to-end resource management which spans the host operating system, host communication subsystem and ATM networks. The research is motivated by *i)* the need to map QoS requirements from the ODP layer to specific resource modules in a clean and efficient manner; *ii)* the need to resolve multimedia synchronization needs of multiple related ODP streams [23]; and *iii)* the need to provide suitable communication protocol support for multimedia services being developed at Université Pierre et Marie Curie.

3.10 End system QoS framework

At Washington University, Gopal and Purulkar [72] have developed a QoS framework for providing QoS guarantees within the end-system for networked multimedia applications. There are four components of the Washington University end system QoS framework as illustrated in Fig. 11: QoS specification, QoS mapping, QoS enforcement and protocol implementation. QoS specification is at a high level and uses a small number of parameters to allow applications greater ease in specifying their flow requirements. Based on QoS specification, QoS mapping operations derive resource requirements for each end-to-end application session. Important system resources considered in [72] include the CPU,

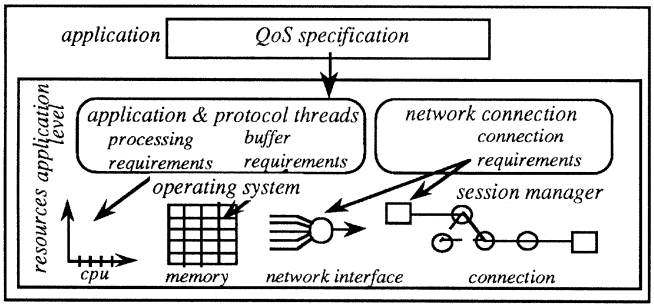

Fig. 11. End system QoS framework

memory and network. The third component of the framework is QoS enforcement. QoS enforcement is mainly concerned with providing real-time processing guarantees for media transfer. A real-time upcall (RTU) facility [81] has been developed for structuring protocols. RTUs are scheduled using a rate-monotonic policy [12] with delayed preemption that takes advantage of the iterative nature of protocol processing to reduce context-switching overhead and increase end-system scheduling efficiency. The final component of the framework is an application-level protocol implementation model. Protocol code is structured as RTUs with attributes that are derived from high-level specifications by QoS mapping operations.

4 Comparison

In this section, we present a simple qualitative comparison of the QoS architectures survey in Sect. 3. We use the elements of the generalized QoS framework (described in Sect. 2) as a basis for the comparison summarized in Table 1.

5 Discussion

All QoS architectures surveyed in Sect. 3 consider QoS specification (e.g., services contracts, flow specs, and service and traffic classes, etc.) to be fundamental in capturing application-level QoS requirements. Although there is a broad consensus on the need for a *flow spec* which captures quantitative performance requirements, there exist two schools of thought on what it should be. On the one hand, XRM and ATM [53] solutions are based on a flow spec that is made up of one or two QoS parameters that identify a traffic class and an average bandwidth. On the other hand, the Tenet, QoS-A and OMEGA architectures adopt a multivalued flow spec (cf. RFC1633, ST-II, RSVP, HieTS). Although both of these proposals seem similar, philosophically they are rather different in practice. The COMET group [28] argues that, by limiting a flow spec to a set of well-defined services in the end-system and traffic classes in the network, complexity in the end-system and network is more manageable. In contrast, Tenet, QoS-A and OMEGA architectures consider such an approach unnecessarily limiting. These groups argue that, by defining a set of discrete QoS classes, applications may be unduly constrained to conform to a QoS class which may not meet the desired application-level QoS requirements.

Level of service (Sect. 2.2) expresses the degree of certainty that the QoS levels specified in a flow spec will be honored. Each architecture offers a different set of services to applications. For example, the Washington University QoS Framework supports three application classes to which it maps application-level flows. These include: i) *an isochronous class*, which is suitable for continuous media flows; ii) *a burst class*, which is appropriate for bulk data transfer; and iii) *a low-delay class*, which is suitable for applications that require a small response time such as an RPC request. The Washington QoS Framework assumes that all applications fall into one of these three general application classes. While all architectures provide services based on both hard (i.e., guaranteed service) and soft (i.e., best effort) QoS guarantees, it is difficult to determine which set optimally covers the application base. Additional services found in the literature include the predicted service (IETF), statistical service (Tenet, XRM and Heidelberg) and the available bit rate service (ATM Forum).

With the exception of the IETF work (which uses RSVP maintained state), all architectures advocate connection-oriented or 'hard-state' solutions to network-level QoS provision; that is, hard state couples path establishment and resource reservation. Work in the IETF on an integrated services architecture (using RSVP and IPv6 flows) described in Sect. 3.4 assumes that network-level QoS guarantees can be built using a 'soft-state' approach; that is, no explicit connection is established but flows traverse intermediate routers on paths that are temporarily (i.e., network state is timed out and periodically refreshed) established. In this instance, path establishment and resource reservation are decoupled. It is argued that a soft-state approach provides better scalability, robustness, and eradicates the round-trip call setup time found in connection-oriented approaches. In [66], Turner suggests a hybrid approach called *ATM-soft*, which benefits from the use of soft state in a native ATM environment. It is still too early to determine which approach is more suitable for future QoS architectures, given the need to support both high-end (e.g., telesurgery and time-critical applications) and low-end (e.g., video conferencing and audio tools) multimedia applications.

Commonalities exist between QoS control and management strategies found in the end-system and network: e.g., admission control, resource management, scheduling mechanisms. The extent to which network-level QoS mechanisms are applicable in the end-systems (or vice versa) remains an open issue. End-system and network devices can be modelled in a similar way: the only real difference is the overall goal that end-system or network devices are set to achieve. For example, the XRM models the end-system as a virtual switch [28] and a set of configurable multimedia devices based on a DAN architecture [16]. It is evident that commonalities exist between scheduling strategies found in switches/routers and end-system operating systems (e.g., fair-share techniques can be found in the end-system and network switches/routers). This seems encouraging in the first instance. A counter argument, however, is that end-systems have fundamentally different scheduling goals than routers and switches. End-systems schedule a wide variety of both isochronous (e.g., continuous media flows) and asynchronous (e.g., RPCs) work, whereas switches and routers

Table 1. Comparison of QoS architectures

QoS framework elements	QoS provision			QoS control					QoS management	
QoS Models	QoS Mapping	Flow Spec and Resource allocation	Adm. Control/ Coordination	E2E[a] Flow Scheduling	Flow Shaping	Flow Control	QoS Filtering	Flow Synchronization	Monitoring/ Alerts	QoS Maintenance
XRM [28]	EN	EN	(E)N	(E)N	-	N	-	-	N	-
QoS-A [68]	EN	E(N)	EN	E(N)	E	(E)	(E)N	E	EAD	ENRS
ISO [67]	(E)(N)	EN	EN	-	-	-	-	-	EN	EN
Heidelberg [71]	(E)N	EN	EN	E(N)	(E)	(N)	N	-	ED	ERS
TINA [76]	(E)	(N)	N	-	-	-	-	(N)	(N)	-
IETF [62]	EN	-	E	-	-	-	-	-	EN	ENR
Tenet [84]	EN	N	N	N	N	(E)	N	-	ED	ERS
MASI [77]	E(N)	E(N)	E	E	-	-	-	E	E	E
OMEGA [70]	E,(N)	E,(N)	E(N)	E(N)	E	E	-	-	E	ER
WashU [72]	E	E		E	E	-	-	-	-	ER

[a] The term "E2E coordination" refers to the coordination of end-system and network resources for flows. This could be provided by a resource reservation protocol (e.g., RSVP [48]), connection setup protocol (e.g., RCAP [51]) or signaling protocol (e.g., UNI 4.0 [53]).

The legend for the comparison table is as follows:

-	"not addressed"
E/N	"addressed in detail in the end-system/network"
(E)/(N)	"mentioned only in the end-system/network"
R	"QoS renegotiation addressed in detail"
(R)	"QoS renegotiation mentioned only"
S	"QoS scaling addressed in detail"
D	"QoS degradation addressed in detail"
(D)	"QoS degradation mentioned only"
A	"QoS availability in detail"

are mainly involved with switching/routing of cells/packets. This means that in the end-system application execution times (i.e., a quantum [34] of work as illustrated in Fig. 2) can vary widely (e.g., uncompressing a video flow is computationally more intensive than displaying video to a screen). In contrast, switch and router schedulers are generally moving packets/cells from queues to ports or vice versa and are optimized for that task. Therefore, techniques resident in switches (such as HRR [58]) may be inappropriate in host operating systems.

6 Conclusion

In this paper, we have argued that multimedia systems designers should adopt an end-to-end approach to meet application-level QoS requirements. To meet this challenge we have proposed a generalized QoS framework that is motivated by five design principles; that is, the principles of transparency, integration, separation, multiple time scales and performance. Elements of our generalized framework include QoS specification and static and dynamic QoS management. We have summarized and evaluated key research in QoS architectures for distributed systems and discussed some of the issues that emerged during a comparison of the existing QoS architectures. The work presented in this survey represents a growing body of research which is laying the foundations for future QoS programmable multimedia platforms [28, 91]. While the area of QoS research in multimedia networking is mature [1], work on QoS architecture remains in its early stages of development with no substantial implementation results having been published to validate the approach. Given that, the work presented in this paper contributes towards a qualitative understanding of the key principles, services and mechanisms needed to build QoS into distributed multimedia systems.

References

1. Keshav S (1993) Report on the Workshop on Quality of Service Issues in High-Speed Networks. ACM Comput Commun Rev 22 (1): 6–15
2. Kurose JF (1993) Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks. ACM Comput Commun Rev 23 (1): 6–15
3. Vogel A, Bochinatin G von, Disallow R, Geckos J, Kerherv B (1994) Distributed Multimedia Applications and Quality of Service – A Survey. IEEE Multimedia, April 1994
4. Miloucheva I (1995) Quality of Service Research for Distributed Multimedia Applications. In: ACM Pacific Workshop on Distributed Multimedia Systems
5. Partridge C (1990) A Proposed Flow Specification, Internet Request for Comments, No. 1363. Network Information Center, SRI International, Menlo Park, Calif., September 1990
6. Saltzer J, Reed D, Clark D (1984) End-to-end Arguments in Systems Design. ACM Trans Comput Syst 2 (4)
7. Tennenhouse DL (1990) Layered Multiplexing Considered Harmful, Protocols for High-Speed Networks. Elsevier, North-Holland
8. Clark D, Tennenhouse DL (1984) Architectural Consideration for a New Generation of Protocols. In: Proc. ACM SIGCOMM'90, Philadelphia, Pa.

9. Bansal V, Siracusa RJ, Hearn JP, Ramamurthy, Raychaudhuri D (1995) Adaptive QoS-based API for Networking. In: Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, April 1993
10. Kelly FP (1993) On Tariffs, Policing and Admission Control for Multiservice Networks. Proc. Multiservice Networks '93, Cosener's House, Abingdon, July 1993, and Internal Report, Statistical Laboratory, University of Cambridge, UK
11. Liu C, Layland J (1973) Scheduling Algorithms for Multiprogramming in Hard Real Time Environment, J ACM
12. Stankovic et al. (1995) Implications of Classical Scheduling Results for Real-Time Systems, IEEE Comput (Special Issue on Scheduling and Real-Time Systems)
13. Jacobson V (1994) VAT: Visual Audio Tool, vat manual pages, Feb 1993
14. Kanakia H, Mishra P, Reibman A (1993) An Adaptive Congestion Control Scheme for Real Time Packet Video Transport. In: Proc. ACM SIGCOMM'93, San Francisco, Calif., October 1993
15. Pacifici G, Stadler R (1995) An Architecture for Performance Management of Multimedia Networks. In: Proc. IFIP/IEEE International Symposium on Integrated Network Management, Santa Barbara, June 1995
16. Hayter M, McAurely D (1991) The Desk Area Network, ACM Oper Syst Rev
17. Tennenhouse DL, Adam JF, Carver D, Houh HH, Ismert M, Linblad CJ, Stasior W, Wetherall D, Bacher D, Chang T (1994) A Software-Oriented Approach to the Design of Media Processing Environments. In: Proc. IEEE International Conference on Multimedia Computing and Systems, Boston, Mass.
18. Little TDC, Ghafoor A (1990) Synchronisation Properties and Storage Models for Multimedia Objects. IEEE J Sel Areas Commun (3): 229–238
19. Escobar J, Deutsch D, Partridge C (1992) Flow Synchronisation Protocol. In: IEEE GLOBECOM'92, Orlando, Fl., June 1992
20. Hutchison D, Coulson G, Campbell A, Blair G (1994) Quality of Service Management in Distributed Systems. In: Sloman M (ed) Network and Distributed Systems Management, Chapter 11, Addison Wesley, Reading, Mass.
21. APM Ltd (1991) ANSAware 3.0 Implementation Manual. APM Ltd, Poseidon House, Castle Park, Cambridge CB3 0RD, UK
22. Open Software Foundation (1992) Distributed Computing Environment. 11 Cambridge Center, Cambridge, MA 02142, USA
23. International Standards Organization (1992) ODP Draft recommendations X.903: basic reference model of open distributed processing. ISO/IEC JTC1/SC21/WG7
24. Coulson G, Blair GS, Davies N, Williams N (1992) Extensions to ANSA for Multimedia Computing, Comput Networks ISDN Syst 25(11) 305–23
25. Nicolaou C (1990) An Architecture for Real-Time Multimedia Communication Systems. IEEE J Sel Areas Commun 8 (3)
26. Hazard L, Horn F, Stefani JB (1993) Towards the Integration of Real-Time and QoS Handling in ANSA. CNET Report CNET.RC.ARCADE.01
27. Guangxing (1994) A Model of Real-Time QoS for ANSA. Technical Report APM.1151.00.04, APM Ltd, Cambridge, UK
28. Lazar AA, Bhonsle S, Lim KS (1994) A Binding Architecture for Multimedia Networks. In: Proceedings of COST-237 Conference on Multimedia Transport and Teleservices, Vienna, Austria, November 1994
29. Bulterman DC, Liere R van (1991) Multimedia Synchronisation and UNIX. In: Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video. Springer Verlag, Berlin Heidelberg New York
30. Leslie IM, McAuely D, Mullender SJ (1993) Pegasus – Operating Systems Support for Distributed Multimedia Systems, Oper Syst Rev 27 (1)
31. Govindan R, Anderson DR (1991) Scheduling and IPC Mechanisms for Continuous Media. In: Thirteenth ACM Symposium on Operating Systems Principles, Asilomar Conference Center, Pacific Grove, California, SIGOPS, Vol 25, pp 68–80
32. Jeffay K (1993) The Real-Time Producer/Consumer Paradigm: A Paradigm for Construction of Efficient, Predictable Real-Time Systems. In: Proc. 1993 ACM/SIGAPP Symposium on Applied Computing, Indianapolis, Ind.
33. Tokuda H, Kitayama T (1993) Dynamic QoS Control Based on Real-Time Threads. In: Proc. Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster University, Lancaster LA1 4YR, UK
34. Coulson G, Campbell A, Robin P (1995) Design of a QoS Controlled ATM-Based Communication System in Chorus, IEEE J Sel Areas Commun (Special Issue on ATM LANs: Implementation and Experiences with Emerging Technology)
35. Jeffay K, Bennett D (1995) A Rate-Based Execution Abstraction For Multimedia Computing. In: Proc. Fifth International Workshop on Network and Operating Systems Support for Digital Audio and Video, Durham, NH, April 1995
36. Danthine A, Baguette Y, Leduc G, Leonard L (1992) The OSI 95 Connection-Mode Transport Service - Enhanced QoS. In: 4th IFIP Conference on High-Performance Networking, University of Liège Belgium
37. Wolfinger B, Moran M (1991) A Continuous Media Data Transport Service and Protocol for Real-time Communication in High-Speed Networks. In: Second International Workshop on Network and Operating System Support for Digital Audio and Video, IBM ENC, Heidelberg, Germany
38. Feldmeier D (1993) A Framework of Architectural Concepts for High Speed Communication Systems. Computer Communication Research Group, Bellcore, Morristown
39. Doeringer W, Dykeman D, Kaiserswerth M, Meister B, Rudin H, Williamson R (1990) A Survey of Light-Weight Transport Protocols for High-speed Networks. IEEE Trans on Commun
40. Chesson G (1988) XTP/PE Overview. In: Proc. 13th Conference on Local Computer Networks, Pladisson Plaza Hotel, Minneapolis, Minn.
41. Clark DD, Lambert ML, Zhang L (1987) NETBLT: A High-Throughput Transport Protocol. Comput Commun Rev 17 (5)
42. Hehmann DB, Herrtwich RG, Schulz W, Schuett T, Steinmetz R (1991) Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System. In: Second International Workshop on Network and Operating System Support for Digital Audio and Video, IBM ENC, Heidelberg, Germany
43. Schulzrinne H, Casner S (1995) RTP: A Transport Protocol for Real-Time Applications. Work in Progress, Internet Draft, <draft-ietf-avt-rtp-05.ps>
44. Campbell AT, Coulson G, Garcia F, Hutchison D (1992) A Continuous Media Transport and Orchestration Service. In: Proc. ACM SIGCOMM '92, Baltimore, Md., August 1992 pp 99–110
45. Keshav S, Saran R (1994) Semantics and Implementation of a Native-Mode ATM Protocol Stack. Bell Labs Technical Memorandum. http://www.cs.att.com/csrc/keshav/nauers.html
46. Topolcic C (1990) Experimental Internet Stream Protocol, Version 2 (ST-II). Internet Request for Comments No. 1190 RFC1190, October
47. Clark DD, Shenker S, Zhang L (1992) Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In: Proc. ACM SIGCOMM'92, pp 14–26, Baltimore, Md., August 1992
48. Zhang L, et. al. (1996) RSVP Functional Specification, Working Draft, draft-ietf-rsvp-spec-10.ps
49. Ferrari D, Verma DC (1990) A Scheme for Real-Time Channel Establishment in Wide-Area Networks, IEEE J Sel Areas Commun 8(3) 368–77
50. Ferrari D, Ramaekers J , Ventre G (1992) Client-Network Interactions in Quality of Service Communication Environments. In: Proc. 4th IFIP Conference on High Performance Networking, University of Liège Belgium
51. Benerjea A, Mah B (1991) The Real-Time Channel Administration Protocol. In: Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg
52. Zhang H, Keshav S (1991) Comparison of Rate-Based Service Disciplines. ACM SIGCOMM
53. ATM Forum (1996) ATM User-Network Interface Specifications, Version 4.0, ATM Forum

54. Shenker S, Clark D, Zhang L (1993) A Scheduling Service Model and a Scheduling Architecture for an Integrated Service Packet Network. Working Draft available via anonymous ftp from parcftp.xerox.com: /transient/service-model.ps.Z.
55. Zitterbart M, Stiller B, Tantawy A (1992) A Model for Flexible High-Performance Communication Subsystems. IEEE J Sel Areas Commun
56. Parekh A, Gallagher RG (1993) A Generalised Processor Sharing Approach to Flow Control in Integrated Service Networks – The Multiple Node Case. In: Proc. IEEE INFOCOM'93, pp.521–530, San Francisco, Calif., April 1993
57. Golestani SJ (1990) A Stop and Go Queueing Framework for Congestion Management. In: Proc. ACM SIGCOMM'90, San Francisco, Calif.
58. Keshav S (1991) On the Efficient Implementation of Fair Queueing, Internetworking: Research and Experiences, Vol. 2, pp 157–173
59. Guerin R, Ahmadi H, Naghshineh M (1991) Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks. IEEE J Sel Areas Commun 9 (7)
60. Cruz R (1991) A Calculus for Network Delay: Part I: Network Elements in Isolation. IEEE Trans Inform Theory 37 (1)
61. Hyman J, Lazar AA, Pacifici G (1990) Real-Time Scheduling with Quality of Service Constraints, IEEE J Sel Areas Commun 9 (7)
62. Braden R, Clark D, Shenker S (1994) Integrated Services in the Internet Architecture: an Overview. Request for Comments, RFC-1633
63. Floyd S (1993) Link-Sharing and Resource Management Models for Packet Networks. Draft available via anonymous ftp from ftp.ee.lbl.gov: link.ps.Z, September 1993.
64. Jain R (1997) Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey. Comput Networks ISDN Syst
65. Dabbous W, Diot C (1995) High Performance Protocol Architectures. Technical Report, INRIA, Sophia Antipolis, France
66. Turner J (1995) ATM-Soft: A Mini-Proposal for ATM Network Control Using Soft State. Technical Note, Washington University
67. International Standards Organization, UK (1995) ISO Quality of Service Framework. ISO/IEC JTC1/SC21/WG1 N9680
68. Campbell AT, Coulson G, García F, Hutchison D, Leopold H (1993) Integrated Quality of Service for Multimedia Communications. In: Proc. IEEE INFOCOM'93, San Francisco, Calif., April 1993, pp 732–739
69. Lazar AA (1992) A Real-time Control, Management, Information Transport Architecture for Broadband Networks. In: Proc. International Zurich Seminar on Digital Communications, pp 281–295
70. Nahrstedt K, Smith J (1996) Design, Implementation and Experiences of the OMEGA End-Point Architecture. Technical Report (MS-CIS-95-22), University of Pennsylvania
71. Volg C, Wolf L, Herrtwich R, Wittig H (1996) HeiRAT – Quality of Service Management for Distributed Multimedia Systems, Multimedia Syst J
72. Gopalakrishna G, Parulkar G (1994) Efficient Quality of Service in Multimedia Computer Operating Systems. Department of computer science, Washington University, Report WUCS-TM-94-04, August 1994
73. Lazar AA (1994) Challenges in Multimedia Networking. In: Proc. International Hi-Tech Forum, Osaka, Japan
74. Campbell AT, Coulson G, Hutchison D (1994) A Quality of Service Architecture. ACM Comput Communic Rev, April 1994
75. OMG (1993) The Common Object Request Broker: Architecture & Specification, Rev 1.3. December 1993
76. TINA-C (1995) The QoS Framework. Internal Technical Report, June 1995
77. Besse L, Dairaine L, Fedaoui L, Tawbi W, Thai K (1994) Towards an Architecture for Distributed Multimedia Application Support. In: Proc. International Conference on Multimedia Computing and Systems, Boston, Mass.
78. Sluman C (1991) Quality of Service in Distributed Systems. BSI/IST21/-/1/5:33, British Standards Institution, UK, October 1991
79. Shenker S, Wroclawski J, Network Element Specification Template. Internet Draft, June 1995
80. Integrated Service Working Group (1995) Slides from IETF meeting 31. ftp://mercury.lcs.mit.edu/pub/intserv
81. Gopalakrishna G, Parulkar G (1995) A Real-time Upcall Facility for Protocol Processing with QoS Guarantees. In: 15th ACM Symposium on Operating Systems Principles
82. Hyman J, Lazar AA, Pacifici G (1992) Joint Scheduling and Admission Control. In: Proc. ACM SIGCOMM '92, Baltimore, Md., August 1992
83. Lazar AA, Ngoh LH, Sahai A (1995) Multimedia Networking Abstraction with Quality of Services Guarantees. In: Proc. SPIE Conference on Multimedia Computing and Networking, San Jose, Calif.
84. Ferrari D (1996) The Tenet Experience and the Design of Protocols for Integrated Services Internetworks. Multimedia Syst J, May 1998
85. Delgrossi L, Halstrinck C, Hehmann DB, Herrtwich RG, Krone J, Sandvoss C, Vogt C (1993) Media Scaling for Audiovisual Communication with the Heidelberg Transport System. In: Proc. ACM Multimedia '93, Anaheim, Calif.
86. Campbell AT, Coulson G (1996) Transporting QoS Adaptive Flows. Multimedia Syst J, April 1998
87. Anderson DR, Herrtwich RG, Schaefer C (1991) SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in the Internet. Internal Report , University of California at Berkeley, Calif.
88. Nahrstedt K, Smith J (1995) The QoS Broker. IEEE Multimedia
89. Vogel A, Bochmann G von, Dssouli R, Gecsei J, Hafid A, Kerherve B (1994) On QoS Negotiation in Distributed Multimedia Application. In: Proc. Protocol for High-Speed Networks
90. Yeadon N, Garcia F, Campbell A, Hutchison D (1994) QoS Adaptation and Flow Filtering in ATM Networks. In: Second International Workshop on Advanced Teleservices and High Speed Communication Architectures, Heidelberg, Germany
91. Campbell AT (1996) Making Multimedia Networks Programmable Workshop on Gigabit Network Technology Distribution, Washington University, St. Louis, Mo.

CRISTINA AURRECOECHA graduated from University of the Basque Country, worked for Telefonica and lectured at the School of Telecommunications Engineering in Bilbao prior to studying at Columbia University. Cristina is a member of the COMET Group at the Center for Telecommunications Research and is a PhD candidate working with Prof. Aurel A. Lazar. Her research interests include the design and implementation of architectures for integrated management of multimedia networks and services. In the past, Cristina has played an active role in the development of the TINA Software Architecture while working as a summer intern with the TINA Consortium. Currently Cristina is investigating new service management APIs while working towards the completion of her thesis at Columbia.

ANDREW T. CAMPBELL joined the E.E. faculty at Columbia as an Assistant Professor in January 1996 from Lancaster University, where he conducted research in multimedia communications as a British Telecom Research Lecturer. Before joining Lancaster University, Dr. Campbell worked for 10 years in industry, focusing on the design and development of network operating systems, communication protocols for packet-switched and local area networks, and tactical wireless communication systems. Dr. Campbell is deeply interested in the confluence of quality of service and networked multimedia systems research. He is a member of the COMET Group at Columbia's Center for Telecommunications Research. The COMET Group's mission is to build open QoS programmable multimedia networks for the 21st Century. He is currently a co-chair of the IFIP Fifth International Workshop on Quality of Service (IWQoS'97).

LINDA HAUW has recently joined Alcatel Telecom Research Division in Marcoussis, France, where she is a member of the TMN Solutions Group working in the area of network management. Before joining Alcatel, Dr. Hauw was a visiting scholar with the COMET Group of the Center for Telecommunications Research at Columbia University, where she worked on multicast service design for ATM-based networks. Dr. Hauw received an M.S. in electrical engineering from Université d'Orsay (1991) and Ph.D. from Université Paris 6 (1994). The subject of her Ph.D. was concerned with quality-of-service architectures and management for high-speed networks. Her current research interests include network management for SDH and ATM networks. In addition, Dr. Hauw is investigating the use Corba and distributed processing techniques in this context.